(2)

ADAPTIVE HYBRID PICTURE CODING

Interim Scientific Report

by

Richard A. Jones
Principal Investigator
University of Arkansas
Department of Electrical Engineering
Fayetteville, Arkansas 72701

July 1988

Prepared for the
Air Force Office of Scientific Research

under
Grant No. 84-0322

ADA200059

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 88-1011
6a. NAME OF PERFORMING ORGANIZATION Univ of Arkansas Dept of Electrical Engineering	6b. OFFICE SYMBOL (If applicable)
7a. NAME OF MONITORING ORGANIZATION AFOSR/NE	
6c. ADDRESS (City, State and ZIP Code) Fayetteville, AR 72701	7b. ADDRESS (City, State and ZIP Code) Bldg 410 Bolling AFB, DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFOSR	8b. OFFICE SYMBOL (If applicable) NE
9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-84-0322	
8c. ADDRESS (City, State and ZIP Code) Bldg 410 Bolling AFB DC 20332-6448	

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
61102 F	2305	B3	

11. TITLE (Include Security Classification)
Adaptive Hybrid Picture Coding

12. PERSONAL AUTHOR(S)
Dr. Jones

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final Report	FROM 30Sep84 TO 30Sep88	July 88	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

A novel approach to quantize design through the employment of estimation theory techniques leads to a solution to the problem of quantizing noise corrupted sources. By viewing the quantizer as a general estimation device, a risk function can be defined for predetermined cost functions and minimized to realize a quantizer structure which counteracts the effects of the additive noise.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
Giles	(202) 767-4931	NE

DD FORM 1473, 83 APR   EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE

A Risk Theory Approach to Quantizer Design

Abstract of dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

By

Mark Kevin Cook, B.S., M.S.E.E.,
University of Arkansas, 1980
University of Arkansas, 1983

~~May 1987~~
University of Arkansas

ABSTRACT

A novel approach to quantizer design through the employment of estimation theory techniques leads to a solution to the problem of quantizing noise corrupted sources. By viewing the quantizer as a general estimation device, a risk function can be defined for predetermined cost functions and minimized to realize a quantizer structure which counteracts the effects of the additive noise. The resulting quantizer provides superior performance w.r.t. the predefined cost function over all other quantizer schemes.

After showing that any desired estimation technique (Bayesian, minimax, etc.) can be used, a constrained Bayesian approach is used to develop quantizers for the squared error, absolute error, and uniform error cost functions. A specific example, the problem of i.i.d. Gaussian source corrupted by additive i.i.d. Gaussian noise, is considered. The minimum risk quantizers that result for the above cost functions are presented. It is further shown that as the noise power becomes small (nearly noiseless conditions), the minimum risk quantizer converges to the well-known Max-Lloyd results. In fact, in a noise-free environment, the quantizer requirements are identical.

The theory for designing a minimax risk quantization scheme is presented where the statistics of only one of the sources (clean information source or corruptive noise source) is well defined and the other is uncertain. Once again, for noiseless conditions the risk function is shown to collapse to classical results for both cases.

Results of quantizing a noisy image source with minimum risk and Max-Lloyd quantizers are also presented and compared.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Section I

INTRODUCTION

A considerable amount of attention has been given to the quantization problem [1][3][6][14][16], the result of which has been some rather elegant and useful results. However, these optimal quantization efforts have been restricted to noiseless sources. When the quantizer input is that of a noise corrupted source, the performance may deteriorate. It will be shown in this dissertation that the effects of the noise can be reduced by considering the quantizer as a general estimation device. The problem then becomes one of designing the quantizer such that the average distortion between the source (rather than quantizer input) and the quantizer output is minimum. The problem can be solved by applying a modified Bayesian Risk theory. In particular, since the input is source plus noise, the levels and thresholds may be determined such that the average risk is minimized.

An outline of this report is as follows. The general quantization problem is first presented in Section II, followed by a brief review and examination of previous results in optimal scalar quantization theory. The problem of a source corrupted by independent additive noise prior to quantization is introduced and examined in Section III. Also in Section III, the theoretical derivation of an optimal quantization scheme for this problem is presented which uses estimation theory techniques to minimize the risk associated with quantization of the noisy source. This scheme will assume prior knowledge of the source and noise probability distributions and general statistics. The results are shown to converge to the well known Max-Lloyd results as the noise power decreases.

A minimax estimation approach to the risk analysis of the noisy quantization problem, when the prior knowledge is incomplete, is presented in Section IV. The results indicate a procedure for quantizing a well-defined random source corrupted by noise of uncertain distribution. Further, the process is reversible to yield a quantization approach when the noise distribution is well-defined but the source is not. Once again, the results agree with classical results for noiseless systems. Finally, results and conclusions appear in Section V.

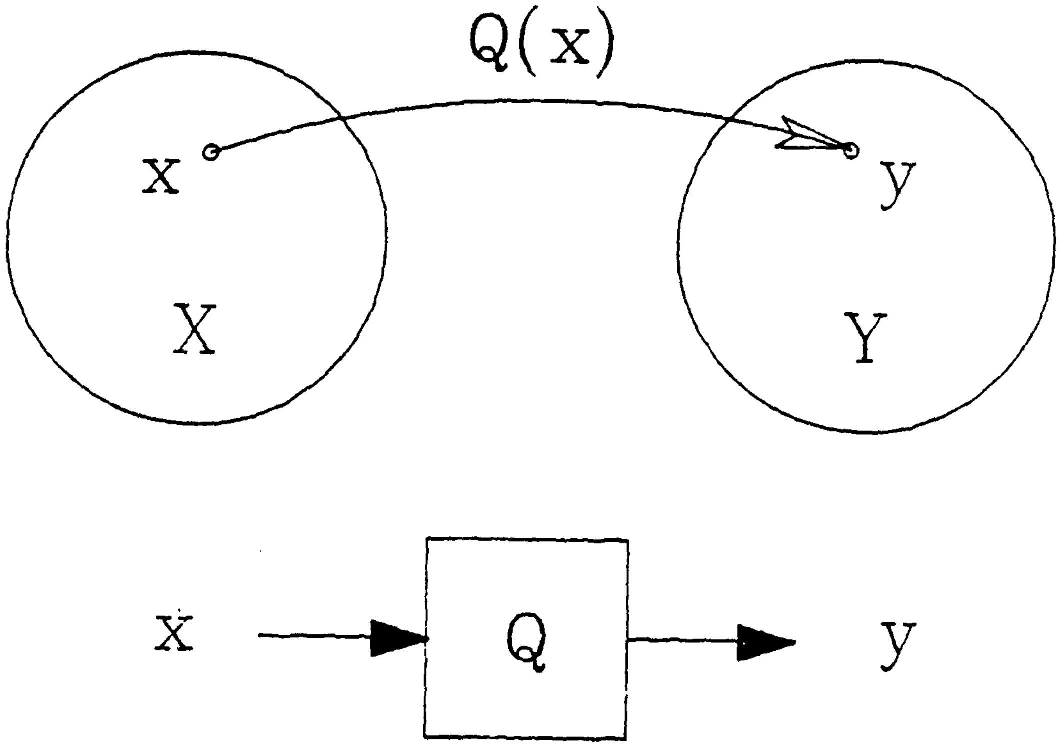
Section II

THE QUANTIZATION PROBLEM

Quantization describes the process of applying a surjective mapping of samples from a large domain space to a smaller, finite range space. This process is schematically illustrated in figure 1. The quantizer may be viewed as a decision function incorporating all decision processes. From simple scalar threshold logic to sophisticated multidimensional parameter estimation, quantization provides data rate reduction and decision responses for digital systems. System performance criteria necessitate a thorough understanding of the underlying concepts of quantization to obtain suitable mappings.

2.1 Quantizer Design

Most of the work in quantizer design begin with the quantizer mapping definition and the additional characteristic of input estimation. That is, a decision is made regarding which region of the domain an input sample lies, and a representative value is determined which best estimates elements of that region. See



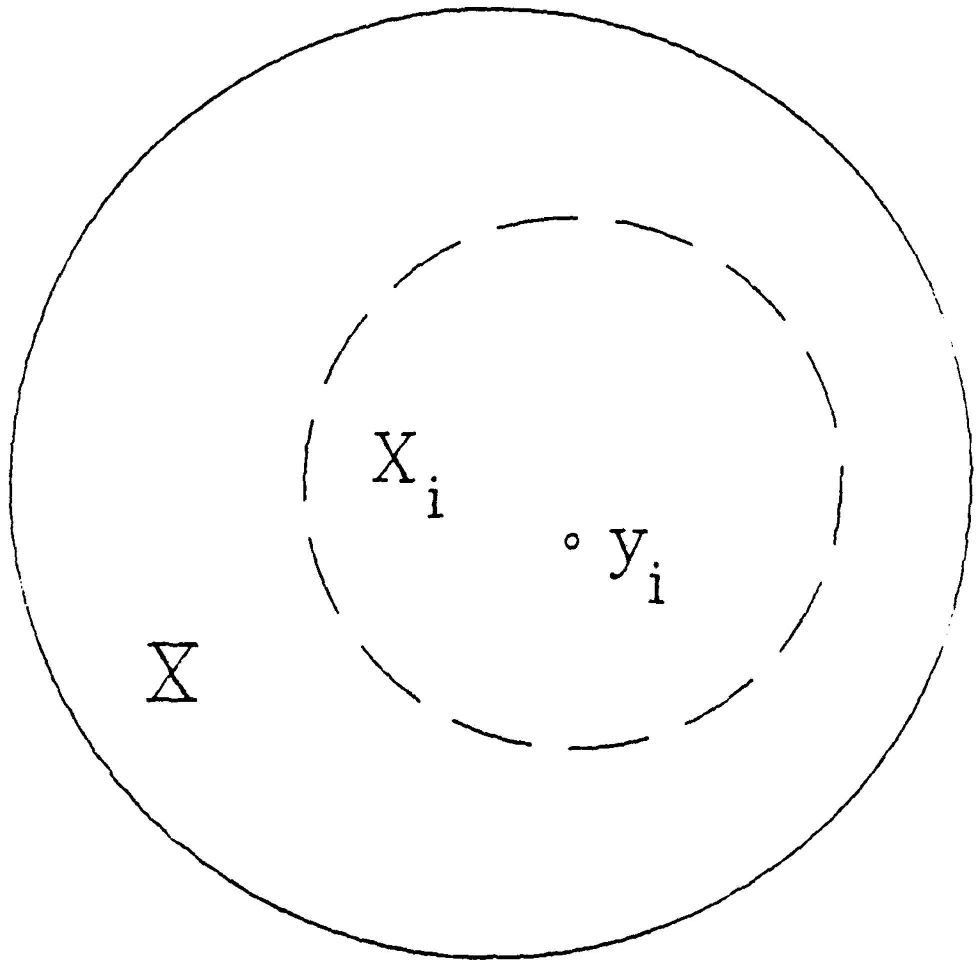
$$Q(x):\quad x \in X \rightarrow y \in Y,\quad Y=\{y_1,\ldots,y_N\}$$

Figure 1

figure 2. Thus possible mappings are restricted to those that provide representative values which are elements of each respective region, and the entire range set is a subset of the input domain. It will be shown that this condition is too restrictive when the source is corrupted by noise prior to quantization.

With the definition of a performance measure, and possibly additional restrictions regarding prior knowledge, analyses have been made which produced efficient quantizer mappings. In this work, the concept of quantization as a combination decision/estimation device will be explored from an estimation theory standpoint to provide additional insight into the fundamental nature of quantization, and the problem of quantizing a noise-corrupted source.

Denote the quantizer domain space $\mathbf{X}$, and the finite range space $\mathbf{Y}$. Then the quantizer mapping, $y = q(x)$, produces $y \in \mathbf{Y}$, $\forall x \in \mathbf{X}$. The design of a quantizer consists of completely specifying $q(x)$ and $\mathbf{Y}$, when $\mathbf{X}$ is known. The mapping is surjective since every element in $\mathbf{Y}$ is a possible outcome of $q(x)$, $\forall x \in \mathbf{X}$. Since $\mathbf{Y}$ is finite, call N the number of elements in $\mathbf{Y}$. The number of elements in $\mathbf{X}$ is possibly infinite and need not be



Classical Quantization

Figure 2

specified, but is observed greater than N if data compression is to be achieved. Define an index set on the elements of Υ so that $\Upsilon = \{y_1, \cdots, y_N\}$. The domain can be partitioned into N disjoint sets,

$$X_i = \{x \mid q(x) = y_i,\ x \in \Xi\} \qquad i=1,\cdots,N$$
$$\bigcup_i X_i = \Xi. \tag{2.1}$$

As mentioned above, it is common practice to require that a representative value y estimate the sample x. To this end, Υ and $q(x)$ are defined so that y closely approximates x. This will require that each subset X_i be a connected subset with $y_i \in X_i$. The most general notion of quantization, known as uniform quantization, partitions the domain into N connected sets of equivalent measure, with y_i the center of X_i. However, it has been known for some time that this does not always produce the best results.

2.2. Optimal Quantization

The well known results of Max [16] and Lloyd [14] on optimum N-level scalar quantizer design have proven to be a good foundation and basis of comparison for new quantization techniques. These are briefly reviewed for purposes of comparison in Section III.

Based upon the requisite of input estimation, a procedure is outlined for designing a quantizer subject to a general fidelity criterion, $f(e=x-y)$, with the goal of minimizing the distortion for an *a priori* p.d.f. $p(x)$ on the input random variable X. The distortion is defined to be the expected value of $f(e)$:

$$D = \sum_{i=1}^{N} \int_{x_i}^{x_{i+1}} f(x-y_i)\,p(x)\,dx \qquad (2.2)$$

where the set $\{x_i\}_{i=1,\ldots,N+1}$ comprise the partitions (or thresholds) of the domain with $x_1=-\infty$ and $x_{N+1}=\infty$. The partition set delineates the X_i subsets of the domain $\mathbf{X}$:

$$X_1 = (x_1, x_2),$$
$$X_i = [x_i, x_{i+1}), \quad 1 < i \le N$$

and y_k represents $q(x)$, $x \in X_k$. The optimization problem is clearly one of determining the partition set $\{x_i\}$ and level set $\{y_i\}$ so that the minimum distortion is achieved. The resulting necessary conditions for optimal quantization are:

$$f(x_i - y_{i-1}) = f(x_i - y_i), \quad i = 2,\cdots,N \qquad (2.3)$$

and

$$\int_{x_i}^{x_{i+1}} f'(x - y_i)\,p(x)\,dx = 0, \qquad i = 1,\cdots,N \qquad (2.4)$$

For the squared-error function, $f(e) = (x-y)^2$, the expressions become:

$$x_i = \frac{y_i + y_{i-1}}{2}, \qquad i = 2,\cdots,N \qquad (2.5)$$

and

$$\int_{x_i}^{x_{i+1}} (x - y_i)\,p(x)\,dx = 0, \qquad i = 1,\cdots,N \qquad (2.6)$$

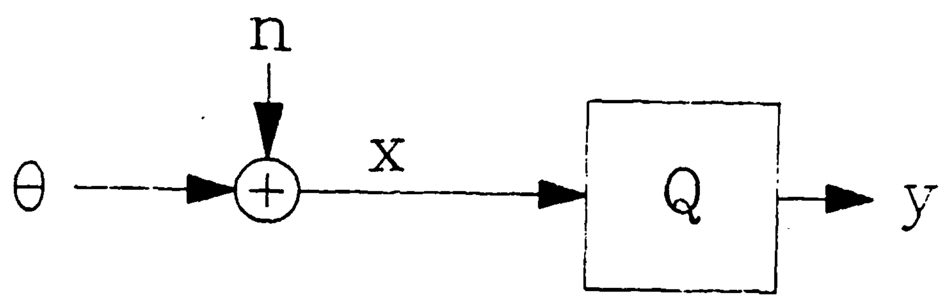
For all but simple distributions, (2.5) and (2.6) are difficult to solve analytically. However, optimal scalar quantizers for several common densities, such as uniform, Gaussian, and Laplacian, have been derived. With the assumption of a uniformly distributed X, they specify the usual criteria for a uniform quantizer. For a Gaussian distributed X, the quantizer mapping is quite different. Max [16] tabulated the results for a quantizer designed with $p(x) \sim N(0,1)$. The characteristics of fine division of the domain for small deviation about the signal mean and coarser division at far deviations are now well known. This is seen as a use of more quantizer levels for regions of high probability than those of low probability, resulting in smaller error for the majority of the signal and lower average distortion. The superiority of the mapping for Gaussian distributions on such sources is obvious when compared to the uniform mapping.

Section III

MINIMUM RISK QUANTIZATION

The problem of quantizing noisy sources, as illustrated in figure 3, has yet to be satisfactorially addressed. An independent identically distributed (i.i.d.) information source which has been corrupted by an (i.i.d.) additive noise source prior to quantization presents a unique challenge to the design of an appropriate quantizer. The usual quantizer design procedure yields the quantizer mapping which produces range elements that best estimate the noisy input samples. This can easily produce unsuitable results.

The problem stems from the normal definition of quantization as an input estimator. A modification of this definition is now proposed. By redefining the goal of the quantizer design procedure to produce a quantizer mapping which yields range elements that best estimate the original information source, the effects of the noise can be greatly reduced. This extension still incorporates input estimation for the noiseless case. The restiction of range elements to membership of the domain subset of



Additive Noise Problem

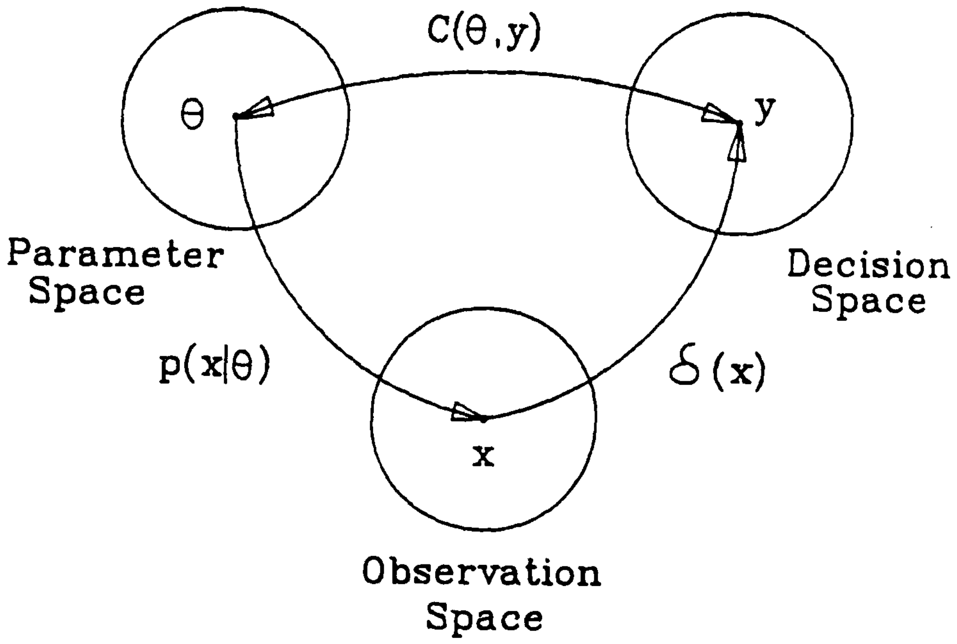
Figure 3

which the preimage is a member, $y_i \in X_i$, is no longer applicable. The reason for this is the noise might corrupt the original signal source such that the best estimate does not lie within the observation set X_i. By realizing the quantizer problem as a general estimation problem, risk theory techniques can be applied for its solution.

3.1 Estimation Theory

The general estimation problem is illustrated in the state space representation of figure 4. The parameter space Θ is related via a probabilistic transition mechanism $p(x|\theta)$ to the observation space $\mathbf{X}$. The decision rule $\delta(x)$ maps the observation space to the decision space Υ. The problem of determining the best estimator y of θ has been addressed in a variety of ways in estimation theory. Perhaps one of the most well known is Bayesian estimation. First a cost function $C(\theta,y)$ over $\Theta \times \Upsilon$ is defined which represents the cost associated with making decision y when the parameter is θ, based on observation x, $\forall\ \theta \in \Theta$, $x \in \mathbf{X}$, $y \in \Upsilon$. It is often appropriate to express the cost function as only the error between the parameter and its estimate.

$$C(\theta, y) = C(e = \theta - y)$$



Estimation Problem

Figure 4

Next the *a priori* probability $p_{\theta}(\theta)$ is used with a specified cost function to form an expression:

$$R = E[C(\theta, y{=}\delta(x))] = \int_{X} \int_{\Theta} C(\theta, y)\, p_{\theta,X}(\theta, x)\, d\theta\, dx \quad (3.1)$$

for the expected cost, termed the risk. The Bayes estimate [22] is said to be that which minimizes the risk.

3.2 Risk Quantization

Analogous to the general estimation problem, the quantizer problem for noise corrupted sources can be expressed similarly. The new definition made here is stated as follows: Quantization is the production of an estimate of a parameter, based on an observation of that parameter, by a value from a lower resolution set than the observation space. It is easily seen that if the parameter to be estimated is the observation (noise-free source as the quantizer input) then the new definition corresponds to the usual definition.

It is desirable to produce the best quantizer for this new definition. The minimum risk criterion is now defined:

$$R^{*} = \min_{Q(x)} \{ E[\ C(\theta, y{=}Q(x)\ ]\ \}$$

$$R^* = \min_{Q(x)} \left\{ \int_X \int_\Theta C(\theta, y)\ p_{\theta, X}(\theta, x)\ d\theta\ dx \right\} \qquad (3.2)$$

where $Q(x)$ corresponds to the decision rule $\delta(x)$. This criterion can be examined through several decision-theoretic techniques, such as Bayesian, minimax, etc., with the appropriate restrictions, to produce the desired quantizer mapping for that particular estimation method. The following estimation procedure is a constrained Bayesian estimation problem because of the restrictions imposed for quantization. These restrictions include the subdivision of the domain into disjoint connected subsets since an estimate is to be produced for each subset rather than the entire domain as per classical Bayesian estimation. Also every element in a specific subset is estimated by a single value for that entire subset instead of the infinite range of values possible with strictly Bayesian estimation.

Using the knowledge (restriction) that quantization requires each value in the quantizer output set, $y_i \in Y \equiv \{ y_1, \cdots, y_N \}$, be produced by an observation within the subdivided input range, $x \in [x_i, x_{i+1}]$, the risk expression becomes:

$$\mathcal{R} = \sum_{i=1}^{N} \left\{ \int_{x_i}^{x_{i+1}} \int_{-\infty}^{\infty} C(\theta, y_i)\ p_{\theta,X}(\theta, x)\ d\theta\ dx \right\} \qquad (3.3)$$

The mapping $Q(x) = \delta(x)$ is then the choice of the set of quantizer levels, $\{y_i\}$, and thresholds, $\{x_i\}$, which minimize this risk. Now the risk can be expressed as:

$$\mathcal{R} = \mathcal{R}_1 + \mathcal{R}_2 + \cdots + \mathcal{R}_N$$

where:

$$\mathcal{R}_i = \int_{x_i}^{x_{i+1}} \int_{-\infty}^{\infty} C(\theta, y_i)\ p_{\theta,X}(\theta, x)\ d\theta\ dx, \qquad (3.4)$$
$$\forall i = 1, \cdots, N$$

The cost function is defined to be nonnegative, so the risk $\mathcal{R}_i$ on each $[x_i, x_{i+1}]$ interval is always a nonnegative quantity. Minimizing the risk:

$$\min\left\{ \mathcal{R} \right\} \Rightarrow \min_{\{x_i\},\{y_i\}} \left\{ \mathcal{R}_k \right\}, \quad \forall k = 1, \cdots, N \qquad (3.5)$$

at a particular threshold x_i:

$$\frac{\partial \mathcal{R}}{\partial x_i} \Rightarrow \frac{\partial}{\partial x_i}\left\{ \mathcal{R}_1 + \mathcal{R}_2 + \cdots + \mathcal{R}_N \right\}$$

$$\Rightarrow \frac{\partial \mathcal{R}_1}{\partial x_i} + \frac{\partial \mathcal{R}_2}{\partial x_i} + \cdots + \frac{\partial \mathcal{R}_N}{\partial x_i} = 0 \qquad (3.6)$$

Now the only intervals affected specifically by x_i are $[x_{i-1}, x_i]$ and $[x_i, x_{i+1}]$, so:

$$\Rightarrow \frac{\partial R_{i-1}}{\partial x_i} + \frac{\partial R_i}{\partial x_i} = 0$$

$$\Rightarrow \frac{\partial}{\partial x_i}\left\{ \int_{x_{i-1}}^{x_i} \int_{-\infty}^{\infty} C(\theta, y_{i-1})\ p_{\theta,X}(\theta, x)\ d\theta\ dx \right\}$$
$$+ \frac{\partial}{\partial x_i}\left\{ \int_{x_i}^{x_{i+1}} \int_{-\infty}^{\infty} C(\theta, y_i)\ p_{\theta,X}(\theta, x)\ d\theta\ dx \right\} = 0 \tag{3.7}$$

The joint density can be rewritten:

$$p_{\theta,X}(\theta, x) = p_{X|\theta}(x|\theta)\ p_\theta(\theta) = p_{\theta|X}(\theta|x)\ p_X(x)$$

providing an expression in terms of the probabilistic transition mechanism $p_{X|\theta}(x|\theta)$ and the *a priori* probability of the parameter space $p_\theta(\theta)$. Now for the case of a system of independent additive noise,

$$p_{X|\theta}(x|\theta) = p_N(n) = p_N(x-\theta),$$

where $p_N(n)$ is the *a priori* probability density of the additive noise source. For conceptual purposes, most expressions will be made in terms of the posterior density $p_{\theta|X}(\theta|x)$, with the knowledge that it can easily be replaced by $p_N(x-\theta)p_\theta(\theta)/p_X(x)$ for calculation simplification. Performing the indicated operations on equation (3.7) results in the necessary condition:

$$\int_{-\infty}^{\infty} C(\theta, y_{i-1})\; p_{\theta|X}(\theta|x_i)\; d\theta = \int_{-\infty}^{\infty} C(\theta, y_i)\; p_{\theta|X}(\theta|x_i)\; d\theta, \qquad \forall i = 1,\cdots,N \tag{3.8}$$

At a particular level y_i:

$$\frac{\partial R}{\partial y_i} \Rightarrow \frac{\partial}{\partial y_i}\left\{ R_1 + R_2 + \cdots + R_N \right\}$$

$$\Rightarrow \frac{\partial R_1}{\partial y_i} + \frac{\partial R_2}{\partial y_i} + \cdots + \frac{\partial R_N}{\partial y_i} = 0 \tag{3.9}$$

this implies that the only risk defined due to y_i is contained within R_i, therefore:

$$\frac{\partial R_i}{\partial y_i} = 0, \qquad \forall i = 1,\cdots,N$$

$$\frac{\partial}{\partial y_i} \int_{x_i}^{x_{i+1}} \left[\int_{-\infty}^{\infty} C(\theta, y_i)\; p_{\theta,X}(\theta, x)\; d\theta \right] dx = 0, \qquad \forall i = 1,\cdots,N \tag{3.10}$$

Equations (3.8) and (3.10) together form the two necessary conditions for minimum risk quantization. These conditions are applicable for all choices of an appropriate cost function. Further reduction in the complexity of (3.8) and (3.10) is not possible until after the choice of cost function is made. Since classically squared error distortion is used, the squared error cost function is selected for further derivation and comparison of the quantizer to well-known classical results, along with absolute error and uniform error cost functions.

3.2.1 Squared Error Cost

When $C(\theta-y)=(\theta-y)^2$, (3.8) becomes:

$$\int_{-\infty}^{\infty} (\theta-y_{i-1})^2 \, p_{\theta|X}(\theta|x_i)\, d\theta = \int_{-\infty}^{\infty} (\theta-y_i)^2 \, p_{\theta|X}(\theta|x_i)\, d\theta \tag{3.11}$$

$$\Rightarrow \quad \int_{-\infty}^{\infty} \theta \, p_{\theta|X}(\theta|x_i)\, d\theta = \frac{y_i+y_{i-1}}{2}, \quad \forall i = 1,\cdots,N \tag{3.12}$$

Using equation (13), the level condition for the squared error cost function is:

$$\frac{\partial}{\partial y_i} \int_{x_i}^{x_{i+1}} \int_{-\infty}^{\infty} (\theta-y_i)^2 \, p_{\theta|X}(\theta|x)\, p_X(x)\, d\theta\, dx = 0 \tag{3.13}$$

After some manipulation this reduces to:

$$y_i = \frac{\int_{x_i}^{x_{i+1}} \left[\int_{-\infty}^{\infty} \theta \, p_{\theta|X}(\theta|x)\, d\theta \right] p_X(x)\, dx}{\int_{x_i}^{x_{i+1}} p_X(x)\, dx}, \quad \forall i = 1,\cdots,N \tag{3.14}$$

where the denominator is the probability of occurrence of an observation within the set $[x_i, x_{i+1}]$. Equations (3.12) and (3.14) can now be used to generate the desired quantizer and are analogous to the Max-Lloyd expressions for squared error distortion. It is interesting to examine these expressions for noiseless conditions. In this case, $X = \theta$, and the squared error cost is the same as the squared error distortion of the Max-Lloyd quantizer. The posterior density $p_{\theta|X}(\theta|x)$ is a delta function at x and the mean is x (likewise the mean of $p_{\theta|X}(\theta|x_i)$ is x_i). Equations (3.12) and (3.14) then are identical to the Max-Lloyd equations for noiseless conditions.

3.2.2 Absolute Error Cost

For the absolute error cost function, $C(\theta-y)=|\theta-y|$, equation (3.8) becomes:

$$\int_{-\infty}^{\infty} \left|\theta-y_{i-1}\right| \; p_{\theta|X}(\theta|x_i)\; d\theta$$

$$= \int_{-\infty}^{\infty} \left|\theta-y_i\right| \; p_{\theta|X}(\theta|x_i)\; d\theta, \qquad \forall i = 1,\cdots,N \tag{3.15}$$

$$\int_{-\infty}^{y_{i-1}} (y_{i-1}-\theta)\ p_{\theta|X}(\theta|x_i)\ d\theta + \int_{y_{i-1}}^{\infty} (\theta-y_{i-1})\ p_{\theta|X}(\theta|x_i)\ d\theta$$

$$= \int_{-\infty}^{y_i} (y_i-\theta)\ p_{\theta|X}(\theta|x_i)\ d\theta + \int_{y_i}^{\infty} (\theta-y_i)\ p_{\theta|X}(\theta|x_i)\ d\theta$$

$$\Rightarrow \int_{-\infty}^{\infty} \theta\ p_{\theta|X}(\theta|x_i)\ d\theta = \frac{y_i+y_{i-1}}{2} \tag{3.16}$$
$$+ \int_{-\infty}^{y_{i-1}} (\theta-y_{i-1})\ p_{\theta|X}(\theta|x_i)\ d\theta + \int_{y_i}^{\infty} (\theta-y_i)\ p_{\theta|X}(\theta|x_i)\ d\theta$$

If the posterior density $p_{\theta|X}(\theta|x_i)$ is symmetric about its mean, it can be shown (see Appendix A) that the mean of the density must lie halfway between y_i and y_{i-1}:

$$\int_{-\infty}^{\infty} \theta\ p_{\theta|X}(\theta|x_i)\ d\theta = \frac{y_i+y_{i-1}}{2}, \quad \forall i = 1,\cdots,N \tag{3.17}$$

but this condition is not necessarily true for nonsymmetric $p_{\theta|X}(\theta|x_i)$. For the absolute error cost function, the level condition is found to be:

$$\frac{\partial}{\partial y_i} \int_{x_i}^{x_{i+1}} \int_{-\infty}^{\infty} \left|\theta-y_i\right|\ p_{\theta|X}(\theta|x)\ p_X(x)\ d\theta\ dx = 0$$

$$\frac{\partial}{\partial y_i} \int_{x_i}^{x_{i+1}} \left[\int_{-\infty}^{y_i} (y_i - \theta)\, p_{\theta|X}(\theta|x)\, d\theta + \int_{y_i}^{\infty} (\theta - y_i)\, p_{\theta|X}(\theta|x)\, d\theta \right] p_X(x)\, dx = 0 \tag{3.18}$$

which reduces to:

$$\int_{x_i}^{x_{i+1}} \left[\int_{-\infty}^{y_i} p_{\theta|X}(\theta|x)\, d\theta \right] p_X(x)\, dx = \frac{1}{2} \int_{x_i}^{x_{i+1}} p_X(x)\, dx \tag{3.19}$$

Equations (3.16) and (3.19) comprise the necessary conditions for a minimum risk quantizer with an absolute error cost function, and (3.17) can be substituted for (3.16) when the posterior density is symmetric about the mean.

3.2.3 Uniform Error Cost

For the uniform error cost function:

$$C(\theta, y_i) = \begin{cases} 1 & |\theta - y_i| > \Delta/2 \\ 0 & |\theta - y_i| \le \Delta/2 \end{cases}$$

the threshold condition is:

$$\int_{-\infty}^{y_{i-1}-\Delta/2} p_{\theta|X}(\theta|x_i)\, d\theta + \int_{y_{i-1}+\Delta/2}^{\infty} p_{\theta|X}(\theta|x_i)\, d\theta = \int_{-\infty}^{y_i-\Delta/2} p_{\theta|X}(\theta|x_i)\, d\theta + \int_{y_i+\Delta/2}^{\infty} p_{\theta|X}(\theta|x_i)\, d\theta \tag{3.20}$$

which results in:

$$\int_{y_{i-1}-\Delta/2}^{y_{i-1}+\Delta/2} p_{\theta|X}(\theta|x_i)\ d\theta = \int_{y_i-\Delta/2}^{y_i+\Delta/2} p_{\theta|X}(\theta|x_i)\ d\theta \qquad (3.21)$$

For $p_{\theta|X}(\theta|x_i)$ densities which are continuous, symmetric about their mean μ, and monotonically decreasing for $\theta > \mu$, then the condition (3.21) (see Appendix B) can also be expressed:

$$\int_{-\infty}^{\infty} \theta\ p_{\theta|X}(\theta|x_i)\ d\theta = \frac{y_i+y_{i-1}}{2}, \quad \forall i = 1,\cdots,N \qquad (3.22)$$

The level condition for the uniform error cost function becomes:

$$\frac{\partial}{\partial y_i}\int_{x_i}^{x_{i+1}}\left[\int_{-\infty}^{y_i-\Delta/2} p_{\theta|X}(\theta|x)\ d\theta + \int_{y_i+\Delta/2}^{\infty} p_{\theta|X}(\theta|x)\ d\theta\right] p_X(x)\ dx = 0 \qquad (3.23)$$

$$\int_{x_i}^{x_{i+1}} \left[\ p_{\theta|X}(y_i-\Delta/2\,|x) - p_{\theta|X}(y_i+\Delta/2\,|x)\ \right] p_X(x)\ dx = 0$$

or

$$p_\theta(y_i-\Delta/2) \int_{x_i}^{x_{i+1}} p_N(x-(y_i-\Delta/2))\, dx$$

$$= p_\theta(y_i+\Delta/2) \int_{x_i}^{x_{i+1}} p_N(x-(y_i+\Delta/2))\, dx \quad (3.24)$$

Equations (3.21) and (3.24) form the necessary conditions for the uniform error cost function and (3.22) replaces (3.21) for the special conditions on the posterior density stated above.

3.3 <u>Gaussian Source + Gaussian Noise Example</u>

The following is an example of the realization of a quantizer satisfying the minimum risk criterion for a specific situation, and is intended to illustrate the application of the general theory developed in the previous section. Consider a case similar to the Max-Lloyd derivation of a minimum squared error distortion quantizer for a source of known Gaussian statistics. Let the source θ be Gaussian with known mean and variance, μ_θ and σ_θ^2 respectively. Likewise let the noise N be Gaussian with a mean and variance of μ_N and σ_N^2.

3.3.1 <u>Squared Error Cost (G.S. + G.N.)</u>

For the squared error cost function, equations (3.12) and (3.14) may be used. Equation (3.12) results in:

$$K_1 x_i + K_2 = \frac{y_i + y_{i-1}}{2}, \qquad \forall i = 1, \cdots N \tag{3.25}$$

and (3.14) becomes:

$$y_i = K_1 \frac{\int_{x_i}^{x_{i+1}} x \, p_X(x) \, dx}{\int_{x_i}^{x_{i+1}} p_X(x) \, dx} + K_2, \tag{3.26}$$

$$\forall i = 1, \cdots, N$$

where:

$$K_1 = \frac{\sigma_\theta^2}{\sigma_\theta^2 + \sigma_N^2}, \qquad K_2 = \frac{\mu_\theta \sigma_N^2 - \mu_N \sigma_\theta^2}{\sigma_\theta^2 + \sigma_N^2}$$

and $p_X(x)$ is Gaussian with mean $\mu_\theta + \mu_N$, and variance $\sigma_\theta^2 + \sigma_N^2$. (See Appendix C).

Equations (3.25) and (3.26) are analogous to the Max-Lloyd equations (2.5) and (2.6), respectively. It is interesting to compare the two sets of conditions for this example. Since the minimum risk quantizer makes use of the additional knowledge of the noise statistics, it is conceivable that the Max-Lloyd quantizer could be designed with an input p.d.f. modeled as the convolution of the source p.d.f. and noise p.d.f. With all of these assumptions for this particular case, the risk quantizer differs from the Max-Lloyd quantizer through the values for K_1 and K_2. Rewriting,

$$K_1 = \frac{\dfrac{\sigma_\theta^2}{\sigma_N^2}}{\dfrac{\sigma_\theta^2}{\sigma_N^2} + 1}, \qquad K_2 = \frac{\mu_\theta - \mu_N \dfrac{\sigma_\theta^2}{\sigma_N^2}}{\dfrac{\sigma_\theta^2}{\sigma_N^2} + 1} \qquad (3.27)$$

The risk and Max-Lloyd quantizers for this example can be compared by examining the problem with large and small noise power. The common measure of noise is given by the signal-to-noise ratio (SNR), defined by the ratio of the signal variance, σ_θ^2, to the noise variance, σ_N^2. From equation (3.27), it is seen that in the case of large SNR, K_1 approaches unity, while K_2 approaches $-\mu_N$, the noise mean. This implies that the larger the SNR, the closer the risk quantizer approaches the Max-Lloyd quantizer, offset by the noise mean. For small SNR, K_1 is small and approaches zero, while K_2 approaches μ_θ, the signal mean. This is intuitively pleasing, since it implies that the quantizer converges to the Max-Lloyd case as expected in the noiseless case. So as the corruptive

TABLE 1

4-Level Quantizers

Gaussian Source + Gaussian Noise Example

(Squared-Error Cost)

					Max-Lloyd		Minimum Risk	
μ_θ	μ_N	σ_θ	σ_N	S/N	x_i	y_i	x_i	y_i
0	0	1	0.10	20	0	0.4550	0	0.4505
					0.9865	1.5179	0.9865	1.5029
0	0	1	0.25	12	0	0.4667	0	0.4393
					1.0118	1.5567	1.0118	1.4653
0	0	1	0.30	10.4	0	0.4727	0	0.4337
					1.0248	1.5769	1.0248	1.4467
0	-2	1	0.10	20	-2	-1.5450	-2	0.4505
					-1.0135	-0.4821	-1.0135	1.5029
0	-2	1	0.25	12	-2	-1.5333	-2	0.4393
					-0.9882	-0.4431	-0.9882	1.4653
0	-2	1	0.30	10.4	-2	-1.5273	-2	0.4337
					-.9752	-0.4231	-0.9752	1.4467

noise gets larger, the estimates from the risk quantizer approach the signal mean, while the estimates from the Max-Lloyd quantizer spread to cover the convolved signal and noise distributions, as do the thresholds for both the Max-Lloyd and risk quantizers. In fact the thresholds of the two quantizers are identical. A comparison of the respective quantizer values at various SNR rates is given in table 1. Note the compression of the levels towards the signal mean at lower SNR for the risk quantizer, where the Max-Lloyd spreads away from it. Also note how the Max-Lloyd quantizer follows the convolved signal as the observation mean changes. The minimum risk, on the other hand, allows only the threshold set for the observation to shift.

A plot of the risk surfaces of 4-level Max-Lloyd and minimum risk quantizers, for the Gaussian source and noise problem with squared error cost, is given in figures 5 and 6. The Max-Lloyd quantizer has been designed with an observation p.d.f. of the convolution of the source and noise p.d.f.s. Note that for zero mean noise, the two quantizers are nearly identical at moderate to high SNR values, yet as the noise power increases with respect to the signal power, the risk associated with each quantizer diverges. It is also apparent from these graphs that the

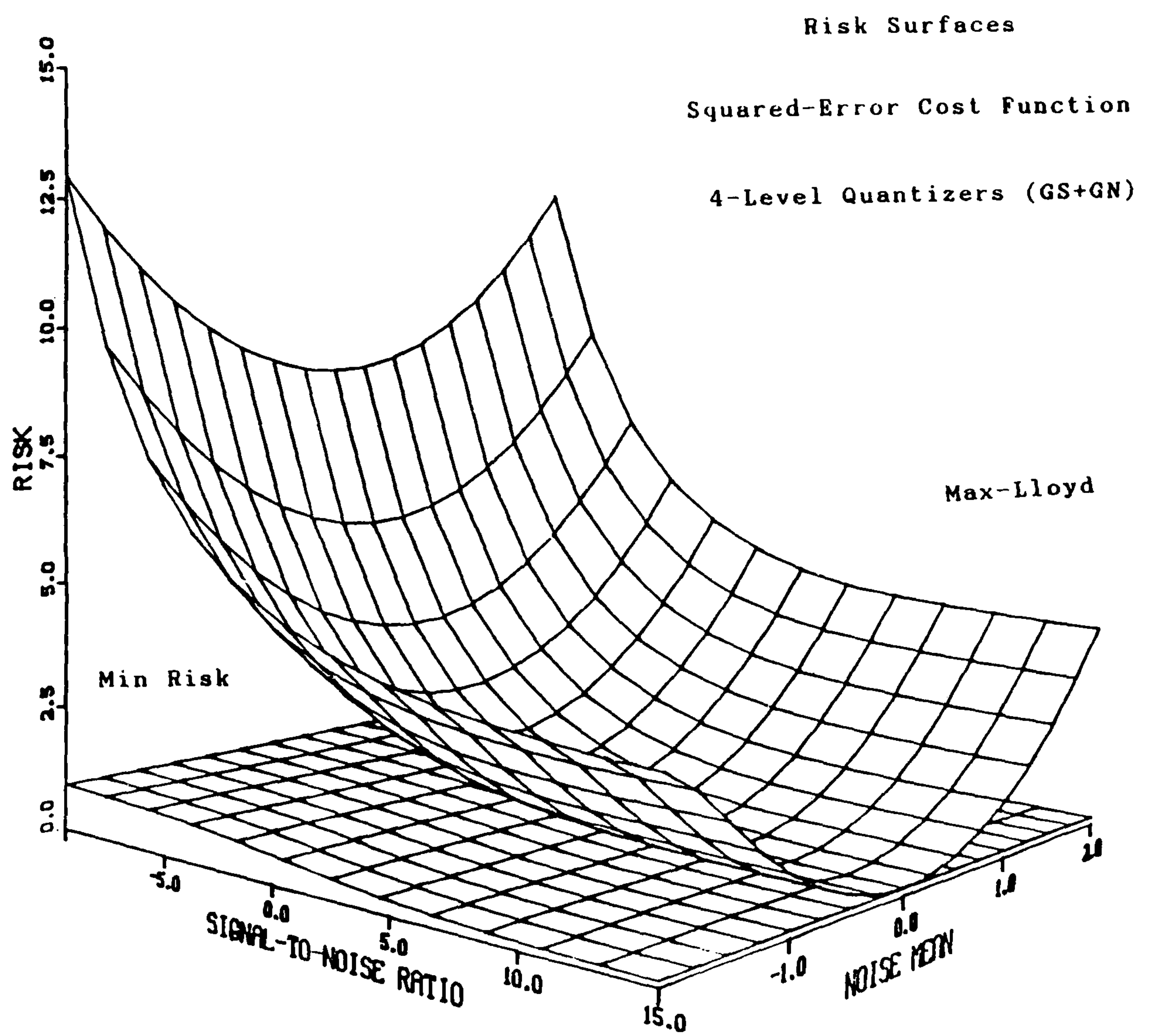

Risk Surfaces
Squared-Error Cost Function
4-Level Quantizers (GS+GN)
Max-Lloyd
Min Risk
RISK
0.0
2.5
5.0
7.5
10.0
12.5
15.0
SIGNAL-TO-NOISE RATIO
-5.0
0.0
5.0
10.0
15.0
NOISE MEAN
-1.0
0.0
1.0
2.0

Figure 5

Quantizer Risk

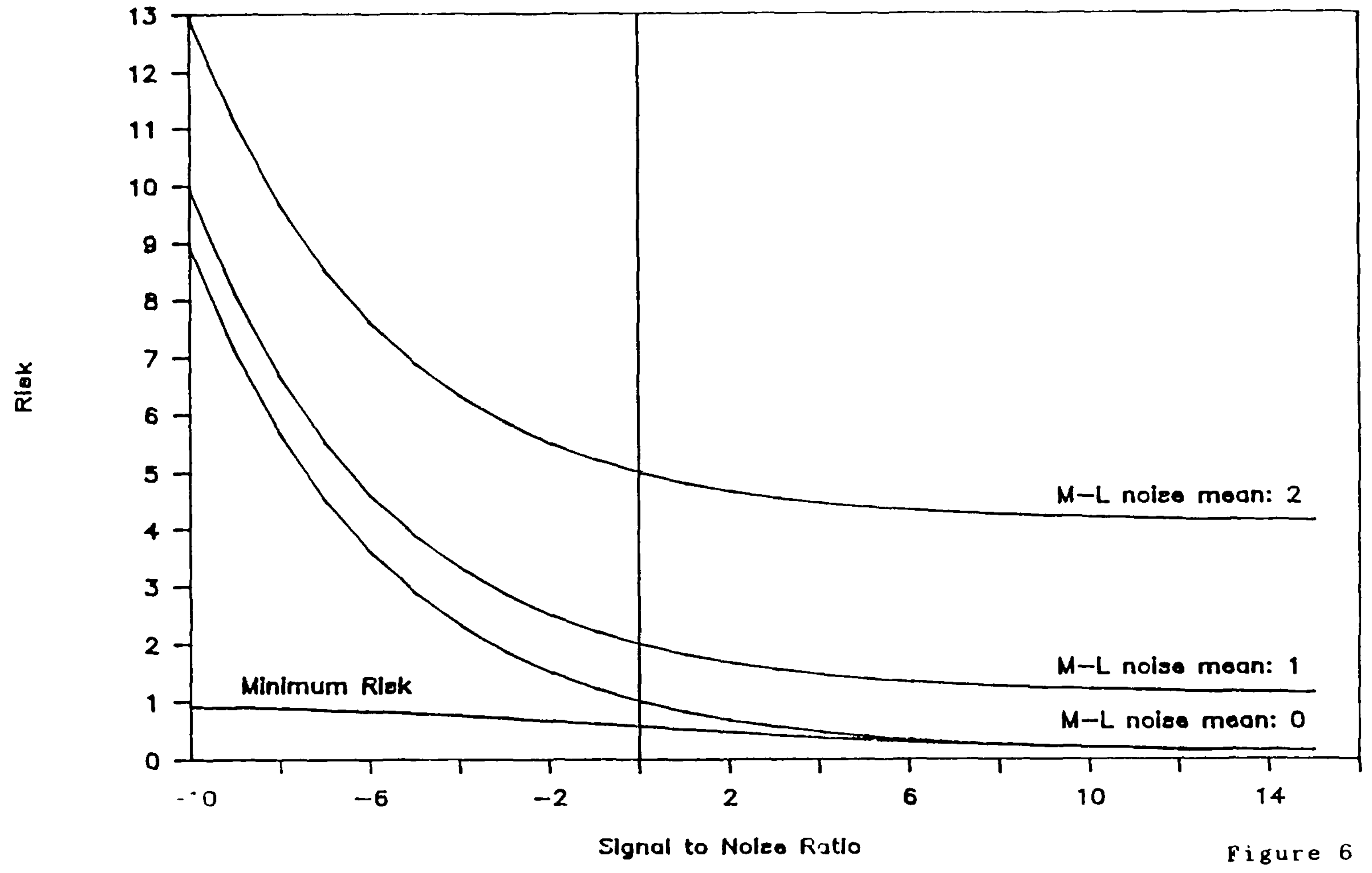


Figure 6

risk for the Max-Lloyd quantizer increases dramatically as the magnitude of the noise mean increases.

For comparison purposes, the risk surfaces are also shown in relation to the risk surface for a uniform quantizer in figures 7 through 10. Note the dramatic increase in risk as compared to the Max-Lloyd and risk quantizers.

3.3.2 Absolute Error Cost (G.S. + G.N.)

For the absolute error cost function the threshold condition for the example is:

$$K_1 x_i + K_2 = \frac{y_i + y_{i-1}}{2}, \qquad \forall i = 1, \cdots N \tag{3.28}$$

and the level condition is:

$$\int_{x_i}^{x_{i+1}} \operatorname{erf}\left[\frac{y_i-(K_1 x+K_2)}{\sqrt{\sigma_\theta^2 \sigma_N^2 \;/\; \sigma_\theta^2+\sigma_\theta^2}}\right] p_X(x)\; dx = \frac{1}{2}\int_{x_i}^{x_{i+1}} p_X(x)\; dx \tag{3.29}$$

where the observation density $p_X(x)$ is Gaussian with mean $\mu_\theta+\mu_N$ and variance $\sigma_\theta^2+\sigma_N^2$. (See Appendix C).

Table 2 shows the results for the Max-Lloyd

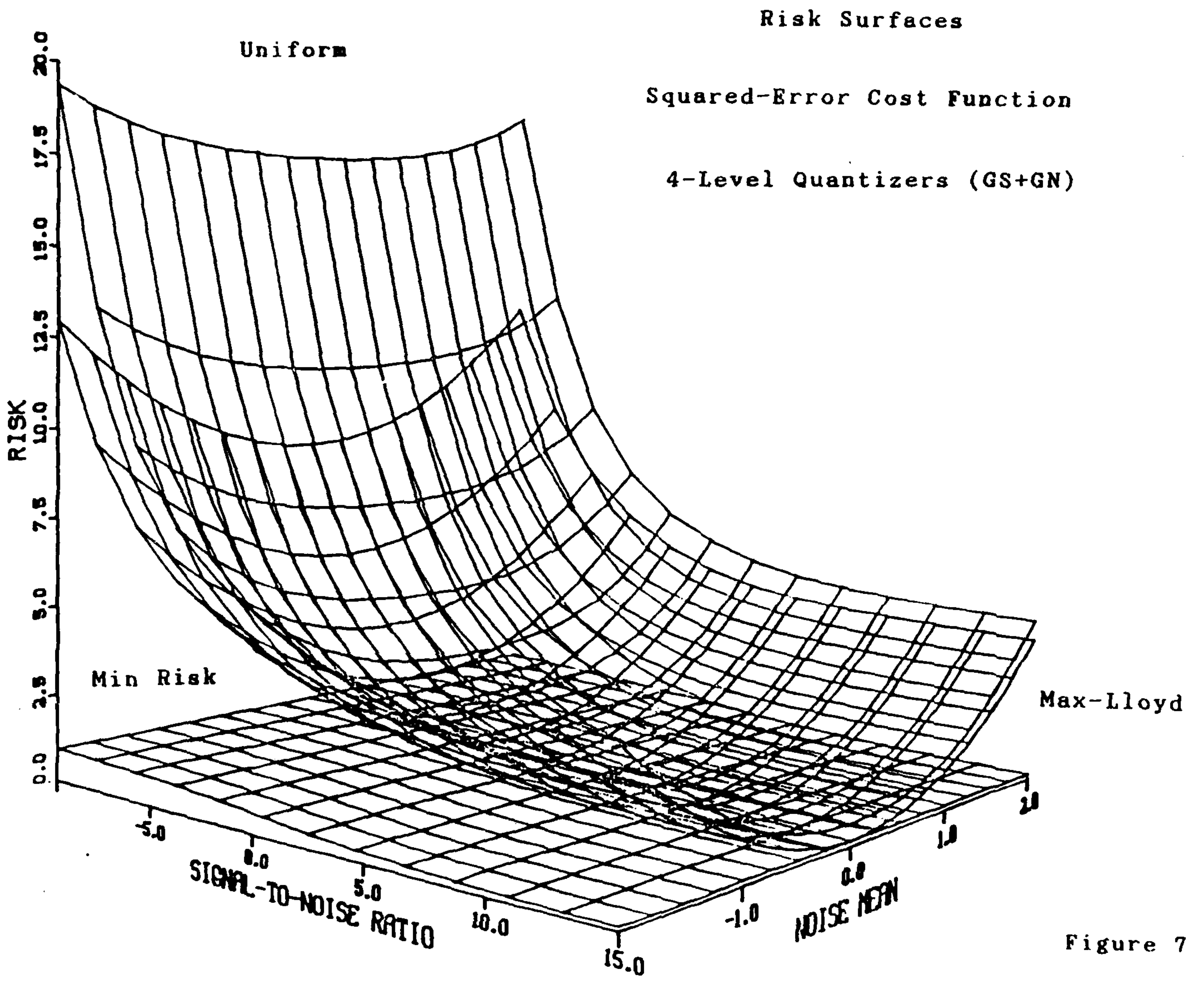
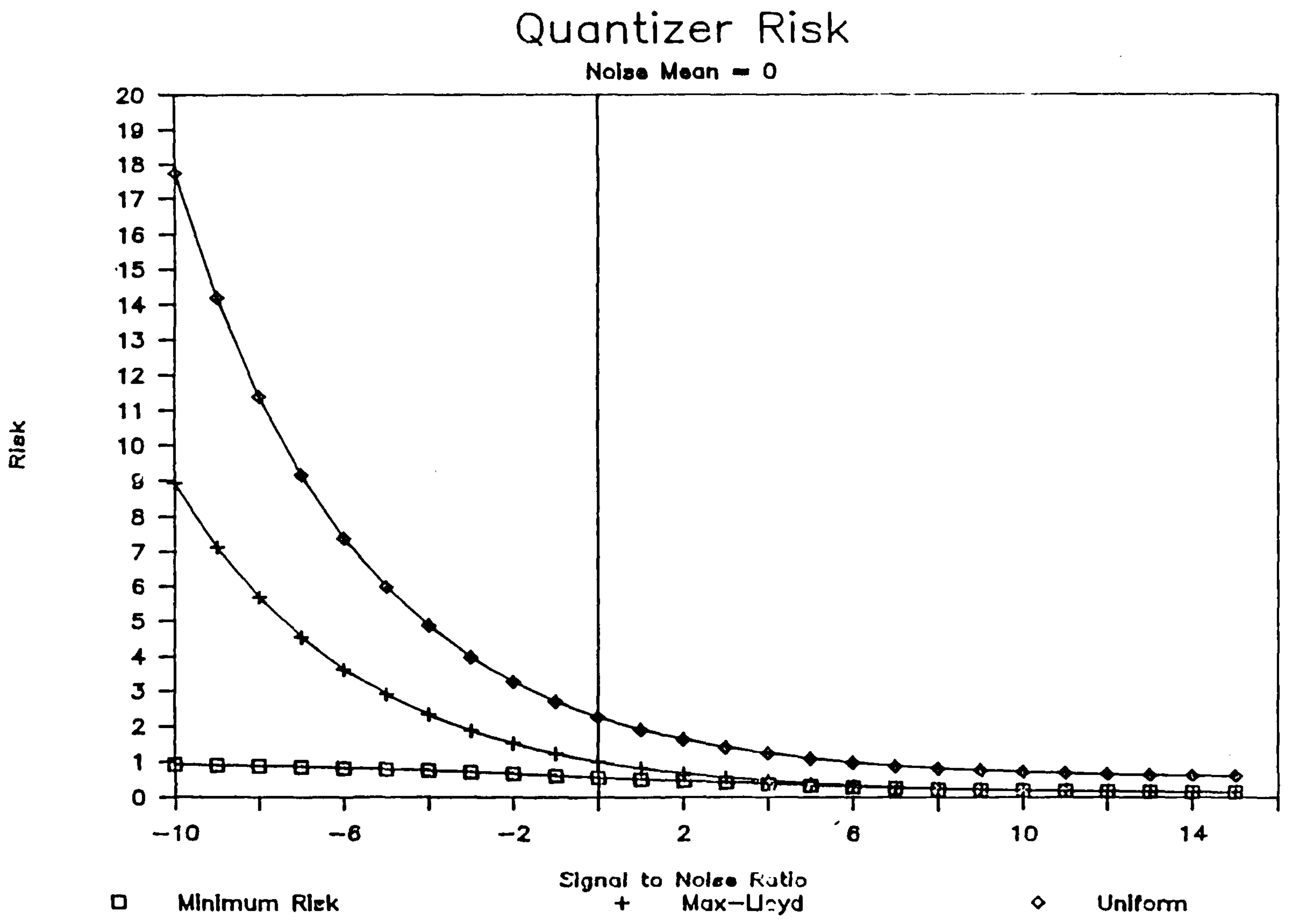

Risk Surfaces
Squared-Error Cost Function
4-Level Quantizers (GS+GN)
Uniform
Min Risk
Max-Lloyd
RISK
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
SIGNAL-TO-NOISE RATIO
-5.0
0.0
5.0
10.0
15.0
NOISE MEAN
-1.0
0.0
1.0
2.0

Figure 7



Quantizer Risk
Noise Mean = 0
Risk
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
−10
−6
−2
2
6
10
14
Signal to Noise Ratio
□ Minimum Risk
+ Max−Lloyd
◇ Uniform

Figure 8

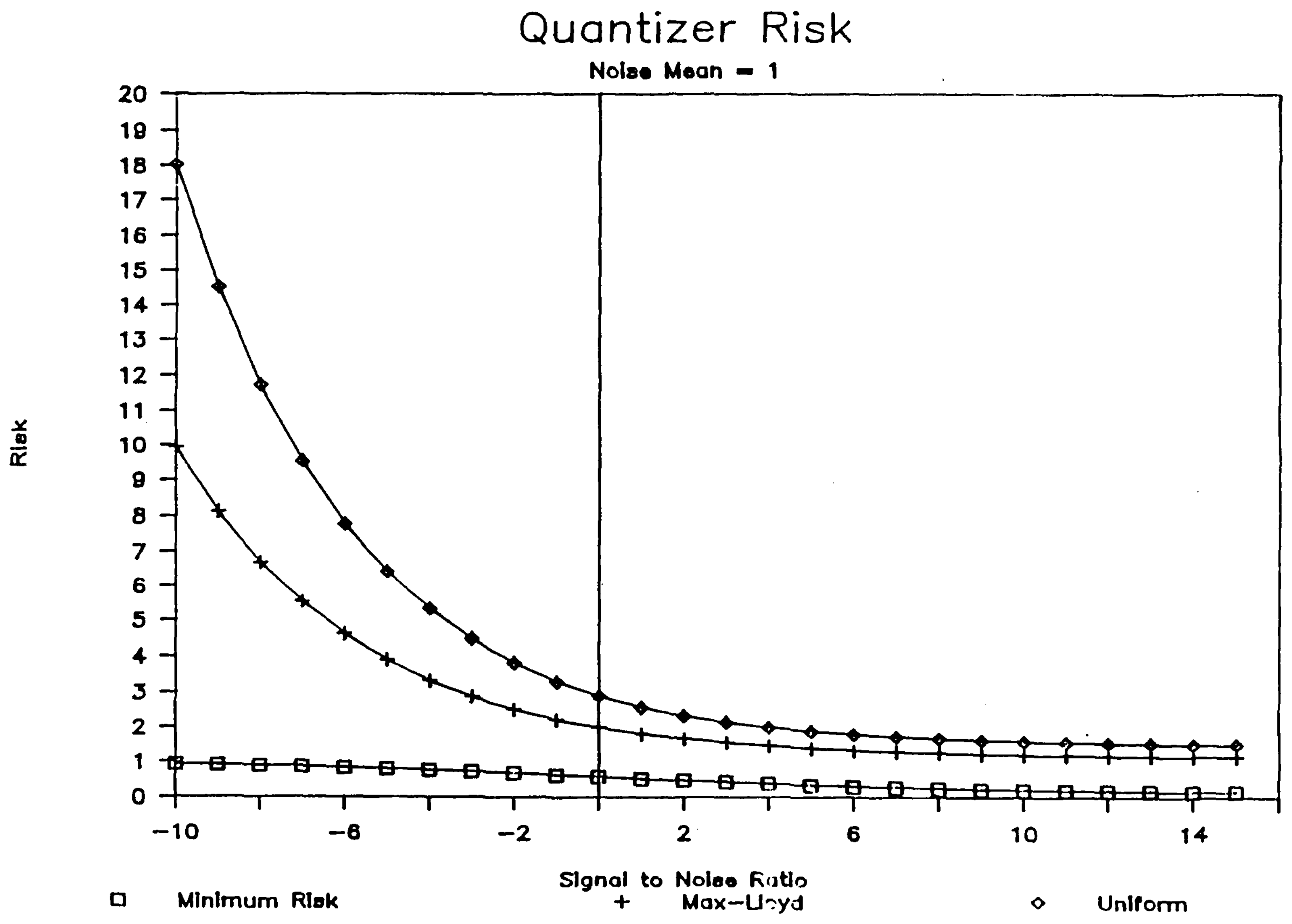

Quantizer Risk
Noise Mean = 1
Risk
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
−10
−6
−2
2
6
10
14
Signal to Noise Ratio
□ Minimum Risk
+ Max−Lloyd
◇ Uniform

Figure 9

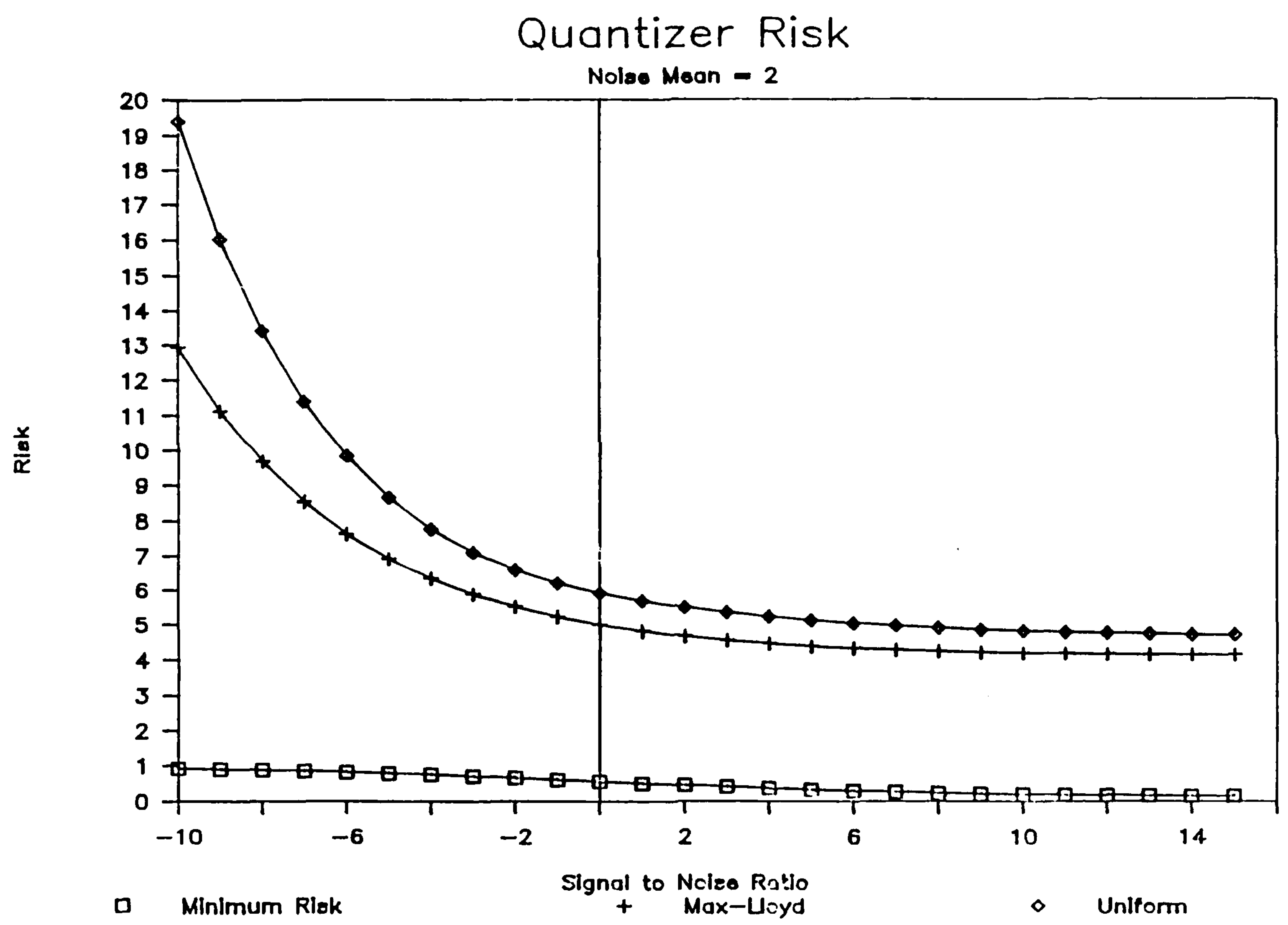

Quantizer Risk
Noise Mean = 2
Risk
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-10
-6
-2
2
6
10
14
Signal to Noise Ratio
□ Minimum Risk
+ Max-Lloyd
◇ Uniform

Figure 10

TABLE 2

4-Level Quantizers

Gaussian Source + Gaussian Noise Example

(Absolute-Error Cost)

$\mu_\theta = \mu_N = 0$

	Max-Lloyd		Minimum Risk	
S/N (dB)	x_i	y_i	x_i	y_i
10	0	0.3962	0	0.3947
	0.8618	1.327	0.9353	1.306
8	0	0.4068	0	0.3921
	0.8845	1.362	0.9789	1.298
4	0	0.4466	0	0.3684
	0.9716	1.496	1.112	1.222
0	0	0.5342	0	0.3143
	1.162	1.790	1.360	1.045

quantizer and minimum risk quantizer when the cost function is the absolute error function between the source and quantization.

3.3.3 Uniform Error Cost (G.S. + G.N.)

For the uniform error cost function the threshold condition for the example is once again,

$$K_1 x_i + K_2 = \frac{y_i + y_{i-1}}{2}, \qquad \forall i = 1, \cdots N \tag{3.30}$$

and the level condition is:

$$\begin{aligned}\exp(\Delta(y_i - \mu_\theta)/2\sigma_\theta^2) & \left\{\operatorname{erf}\left[\frac{x_{i+1} - (y_i + \Delta/2) - \mu_N}{\sigma_N}\right]\right. \\ & \left. - \operatorname{erf}\left[\frac{x_i - (y_i + \Delta/2) - \mu_N}{\sigma_N}\right]\right\} \\ = \exp(-\Delta(y_i - \mu_\theta)/2\sigma_\theta^2) & \left\{\operatorname{erf}\left[\frac{x_{i+1} - (y_i - \Delta/2) - \mu_N}{\sigma_N}\right]\right. \\ & \left. - \operatorname{erf}\left[\frac{x_i - (y_i - \Delta/2) - \mu_N}{\sigma_N}\right]\right\}\end{aligned} \tag{3.31}$$

The complete derivations of the necessary conditions for the Gaussian source with Gaussian noise example presented here may be found in appendix C for all three types of cost functions considered.

Section IV

MINIMAX RISK QUANTIZATION

In the preceeding sections, the problem of quantizing a (i.i.d.) source corrupted by (i.i.d.) additive noise has been examined for the case when the statistics and probability distributions of the source and noise are well defined. The consequent theory developed has produced an attractive approach to quantizer design for such systems, and is equivalent to the classical result for noiseless systems. In this chapter, the problem of an additive noise source will again be examined, but with only incomplete statistics available to the quantizer designer *a priori*.

Specifically, the source statistics and probability distribution will again be assumed known *a priori*. The exact distribution of the noise, however, will be assumed to be unknown, but belonging to a generalized moment constrained class of distributions, C. A risk analysis of the system will be applied using minimax estimation to produce a minimax risk quantization scheme for the system. Further, it will be shown that the analysis is reversible and may be applied to the additive noise problem when the noise probability distribution is well defined, but the knowledge of the source distribution is incomplete.

4.1 Minimax Estimation

It is often likely that the quantizer design must be based on incomplete knowledge of the signal statistics. For example, a problem may arise when a quantizer is needed for a source where the statistics are fairly well known, but various unknown noise distributions can corrupt the source signal. Likewise, the noisy characteristics of a sensor may be well defined, but the exact probability distribution of the source signal to be quantized with the sensor unknown. These types of problems suggest the use of minimax estimation to determine the quantization mapping. Minimax estimation seeks to minimize the maximum possible distortion by the estimator, and is directly applicable to the risk approach taken here for quantizer design.

A minimax estimator t^* is one which guarantees a maximum risk no greater than that for any other estimator [18].

$$\sup_{\theta} \mathcal{R}_{t^*}(\theta) \leq \sup_{\theta} \mathcal{R}_{t}(\theta) \tag{4.1}$$

Likewise, the minimax quantizer will be defined as one which guarantees the minimum maximal risk for all quantizers. Minimax quantization theory has been developed for the classical quantizer problem of noiseless sources by Bath and VandeLinde [4]. The approach developed there will be applied in the risk theory development of an appropriate quantizer for the additive noise problem.

4.2 Minimax Risk Quantizer for Known Source Statistics

As previously mentioned, it may be desirable to produce a minimax quantization scheme for the additive noise problem where the source probability statistics and density function are known *a priori* but with only limited knowledge of the noise statistics. Consider the sequence of independent, identically distributed random variables $\{\theta_k\}$, from the known cumulative probability distribution function (c.p.d.f) F_θ. The source sequence has been corrupted by a noise sequence $\{N_k\}$, with unknown c.p.d.f.s $\{F_{N_k}\}$, to produce the observation sequence $\{X_k\}$ to the quantizer.

The sequence $\{F_{N_k}\}$, while unspecified, are constrained to belong to the set C of all possible c.p.d.f.s with whatever *a priori* information of the noise characteristics is available. The set to be considered here is the set of c.p.d.f.s belonging to the generalized moment constrained class. These distributions are required to have a generalized moment less than or equal to some finite constant. This may be viewed as a restriction of the noise power to be finite.

The minimax risk is defined:

$$\mathcal{R}^* = \min_{q \in \mathcal{Q}_N} \max_{F_{\mathcal{N}} \in C} \mathcal{R}(q, F_\theta, F_{\mathcal{N}}) \qquad (4.2)$$

where $\mathcal{R}(q,F_\theta,F_{\mathcal{N}})$ is the risk for an N-level quantizer q from the set of all possible N-level quantizers $\mathcal{Q}_N$ and the c.p.d.f.s F_θ, $F_{\mathcal{N}}$, described above. The minimax risk quantizer, q^*, is that quantizer which provides the minimax risk $\mathcal{R}^*$. The properties of the minimax risk quantizer, analogous to the minimax quantizer, are:

1) $\forall\ F_{\mathcal{N}} \in C,\ \nexists\ \mathcal{R}_{q^*} > \mathcal{R}^*$

i.e. $\mathcal{R}^*$ is the absolute maximal risk for quantizer q^*.

2) $\nexists\, q \in \mathcal{Q}_N,\ q \neq q^* \ni \max\limits_{F_{\mathcal{N}} \in \mathcal{C}} \mathcal{R}(q, F_\theta, F_{\mathcal{N}}) < \mathcal{R}^*$

i.e. q^* guarantees a maximum risk no greater than the maximum risk of any other quantizer $q \in \mathcal{Q}_N$.

The quantizer mapping, $q(x) \in \mathcal{Q}_N$, maps an $x \in \mathcal{X}_i$ to a quantizer level $y_i \in \mathcal{Y}$, $i=1,\cdots,N$. For the scalar quantizer, $\mathcal{X}_i = [x_i, x_{i+1})$, where $x_1 = -\infty$, $x_{N+1} = \infty$, and the quantizer thresholds comprise the set $\{x_i\}$, $i=1,\cdots,N+1$. Note that the set of quantizers $\mathcal{Q}_N$ does not require the quantizer level y_i be an element of $\mathcal{X}_i$. The quantizers considered will require the observation sequence to be restricted to a specific range, $x_k \in [-V_X, +V_X]$. This restriction represents a practical constraint which often occurs in common systems.

The risk defined in the previous chapter made use of the *a priori* information of the probability density functions $p_\theta(\theta)$ and $p_{\mathcal{N}}(n)$. A similar representation can be given here for the risk due to a particular quantizer mapping:

$$\mathcal{R}(q, F_\theta, F_{\mathcal{N}}) = \int_\Theta \int_{\mathcal{N}} C(\theta, q(x))\, dF_{\mathcal{N}}\, p_\theta(\theta)\, d\theta \qquad (4.3)$$

where the inner integral is a Lebesgue-Stieltjes integral over the sequence of noise c.p.d.f.s $\{F_{\mathcal{N}_k}\}$. The order of integration is changed to yield:

$$\mathcal{R}(q,F_\theta,F_{\mathcal{N}}) = \int_{\mathcal{N}} \int_{\Theta} C(\theta,q(x))\; p_\theta(\theta)\; d\theta\; dF_{\mathcal{N}} \qquad (4.4)$$

to facilitate the minimax operation procedure.

Once again, it is apparent that if the system is noiseless, the risk function is precisely that of the classical quantizer distortion function. The minimax quantizer then is the same as the Max-Lloyd result since the source distribution is assumed known.

$$\mathcal{R}^* = \inf_q \int_\Theta C(\theta,q(x))\; p_\theta(\theta)\; d\theta, \quad \theta = x$$

$$\Longrightarrow \quad \mathcal{R}^* = \inf_q \int_\Theta C(\theta,q(\theta))\; p_\theta(\theta)\; d\theta \qquad (4.5)$$

The cost function used is restricted to some bounded distortion measure, $d(\theta,q(x))$, between the source and quantizer output:

$$C(\theta,q(x)) = \begin{cases} d(\theta,q(x)), & d(\theta,q(x)) \le L \\ L, & d(\theta,q(x)) > L. \end{cases} \qquad (4.6)$$

The distortion measure $d(\theta,q(x))$ is required to be even, continuous, monotonic strictly increasing in $|\theta - q(x)|$, and zero for perfect estimation. The bound on the cost implies no additional penalty for a distortion larger than the limit, L.

The generalized moment constraint is given by some function $\rho(n)$ satisfying the same conditions as those for the distortion function such that:

$$\int_N \rho(n)\; dF(n) \le c. \tag{4.7}$$

The constraint function $\rho(n)$ simply implies that the noise signal power is bounded. The distortion and constraint functions considered here are:

$$\begin{aligned} d(\theta, q(x)) &= |\theta - q(x)|^n \\ \rho(n) &= |n|^m \qquad m \ge n \ge 1. \end{aligned} \tag{4.8}$$

The conditions of bounded cost and those imposed on $d(\theta, q(x))$ and $\rho(n)$ are required for Lagrange minimization of the next section.

The term "loading factor" was given to the expression x_{N+1}/c in the minimax distortion derivation of Bath and VandeLinde [4] for quantizers of noncorrupted sources. This gives a ratio between the bounded distortion and the input signal power constraint. Since the quantizer input of this derivation is source plus noise, the "loading factor" term will be given as:

$$\text{l.f.} = \frac{x_{N+1}}{\left[\sigma_\theta^2 + c\right]^{.5}} \tag{4.9}$$

to reflect the quantizer input signal power. This factor however, will be unneccessary for the bounded distortion measure chosen for this work since the bound is placed by some fixed L instead of the quantizer threshold range. The term is defined here only for future work with distortion functions similar to that of [4].

The problem then, is to determine the minimax risk quantizer q^* for the worst case c.p.d.f., F_N^*, which achieves:

$$\inf_{q \in \mathcal{Q}_N} \sup_{F_N \in C} \mathcal{R}(q, F_\theta, F_N) = \mathcal{R}(q^*, F_\theta, F_N^*). \tag{4.10}$$

The set $\mathcal{Q}_N$ is the set of all symmetric quantizers where the levels are symmetric about the source mean, and the thresholds are symmetric about the observation mean. The set C is the set of c.p.d.f.s belonging to the generalized moment class, and $\mathcal{R}$ is the risk function using the cost function $C(\theta, q(x))$. To determine the minimax risk quantizer, it first necessary to find the maximum risk due to a particular quantizer mapping $q \in \mathcal{Q}_N$, then determine which quantizer provides the minimum of all of the maximum risks. The maximum risk due to any quantizer mapping $q(x)$

may be determined through the following analysis.

4.2.1 Maximum $\mathcal{R}(q, F_\theta, F_N^*)$

It has been shown [4] that the set C is a weak* compact subset of a normed Banach vector space (N.B.V.) of normalized functions of bounded variation. Also with any linear functional on N.B.V.$[0,\infty]$ (such as $\mathcal{R}(q,F_\theta,F_N)$), which is weak* continuous in F_N, then:

$$\forall\, q \in \mathcal{Q}_N,\ \exists\, F_N^* \in C,\ \mathcal{R}^*(q) \ni :$$
$$\mathcal{R}^*(q) = \mathcal{R}(q, F_\theta F_N^*) = \max_{F_N \in C} \mathcal{R}(q, F_\theta, F_N) \qquad (4.11)$$

Furthermore, a method for determining a minimax quantizer via a constrained minimization in a Lagrange multiplier space ($\mathbb{R}^2$) through the use of the Lagrange duality theorem has been developed [4]. The technique will be paraphrased here to determine a minimax risk quantizer.

Let $C' = \{F_N \in \text{N.B.V.}[0,\infty]\}$ where F_N exhibits the following properties:

1) F_N is nonnegative, monotonically nondecreasing,

2) $\int_{\mathcal{N}} dF_{\mathcal{N}} \le 1,$

3) $\int_{\mathcal{N}} \rho(n)\ F_{\mathcal{N}}(n) \le c.$

Define the convex set $\overline{C} = \{ F_{\mathcal{N}} \in$ N.B.V.$[0,\infty] \}$, where $F_{\mathcal{N}}$ satisfies 1). Now define a convex functional

$$G: \text{N.B.V.}[0,\infty] \longrightarrow \mathbb{R}^2,$$

$$G = \begin{bmatrix} \int dF_{\mathcal{N}} - 1 \\ \int \rho\ dF_{\mathcal{N}} - c \end{bmatrix}$$

which represents the necessary constraints for $F_{\mathcal{N}}$ to be a c.p.d.f. and have a generalized moment constraint. This implies:

$$C' = \{ F_{\mathcal{N}} \in \overline{C}:\ G(F_{\mathcal{N}}) \le 0 \}$$

By the Lagrange duality theorem [15], an expression for the maximum risk due to the quantization mapping $q(x)$ may be found:

$$\mathcal{R}^*(q) = \sup_{F_{\mathcal{N}} \in C} \mathcal{R}(q, F_\theta, F_{\mathcal{N}})$$

$$\sup_{F_{\mathcal{N}} \in C} \mathcal{R}(q, F_\theta, F_{\mathcal{N}}) = \min_{\lambda_1, \lambda_2 \ge 0}\ \sup_{F_{\mathcal{N}} \in \overline{C}} \Big\{ \mathcal{R}(q, F_\theta, F_{\mathcal{N}}) - \lambda_1 \Big[\int dF_{\mathcal{N}} - 1\Big] - \lambda_2 \Big[\int \rho\ dF_{\mathcal{N}} - c\Big] \Big\} \qquad (4.12)$$

where the inner maximization is achieved for some worst case distribution, $F^*_N \in \bar{C}$, and the outer minimization by λ^*_1 and λ^*_2. If the cost weighting function is bounded, $R^*(q)$ is finite and equation (4.4) can be used to replace $R(q, F_\theta, F_N)$ of equation (4.12) to yield:

$$R^*(q) = \min_{\lambda_1, \lambda_2 \geq 0} [(\lambda_1 + \lambda_2 c) + B(\lambda_1, \lambda_2)] \tag{4.13}$$

where

$$B(\lambda_1, \lambda_2) = \max_{F_N \in \bar{C}} \int_N I(n)\, dF_N(n) \tag{4.14}$$

$$I(n) = \int_\Theta C(\theta - q(x))\, p_\theta(\theta)\, d\theta - \lambda_1 - \lambda_2 \rho(n) \tag{4.15}$$

The minimization of equation (4.13) need only be considered when $B(\lambda_1, \lambda_2)$ is finite. This will only occur when $I(n)$ is nonpositive. If $I(n)$ is positive at any point, a sequence of noise random variables $\{N_k\}$ with c.p.d.f.s $\{F_{N_k}\}$ could occur such that as k gets large, the integration tends to infinity.

If $I(n)$ is required to be nonpositive, then the maximization of equation (4.14) over all possible $F_N \in \bar{C}$

implies:

$$B(\lambda_1, \lambda_2) = 0. \qquad (4.16)$$

Since F_{N_k} ≡ constant is an element of $\overline{C}$ for $\{N_k\}$, it follows that if $I(n)$ is made nonpositive, the maximum possible integration of equation (4.14) is zero.

The minimax risk for any quantizer $q \in Q_N$ and the specified conditions then reduces to:

$$R^*(q) = \min_{\substack{\lambda_1, \lambda_2 \geq 0 \\ I(n) \leq 0}} (\lambda_1 + \lambda_2 c) \qquad (4.17)$$

A series of plots appear in figures 11 through 14 which depict the function $I(n)$ for various quantizers with $p_\theta(\theta) \sim N(0,1)$, and distortion bounds of $L = 0.25, 0.75$. These show the effects of the distortion bound on the function, and that it is quantizer dependent. Note the discontinuity near the quantizer threshold.

4.2.2 Minimax Risk Quantizer

The minimax risk quantizer q^* then, is that quantizer which produces:

PLOT OF I(N)

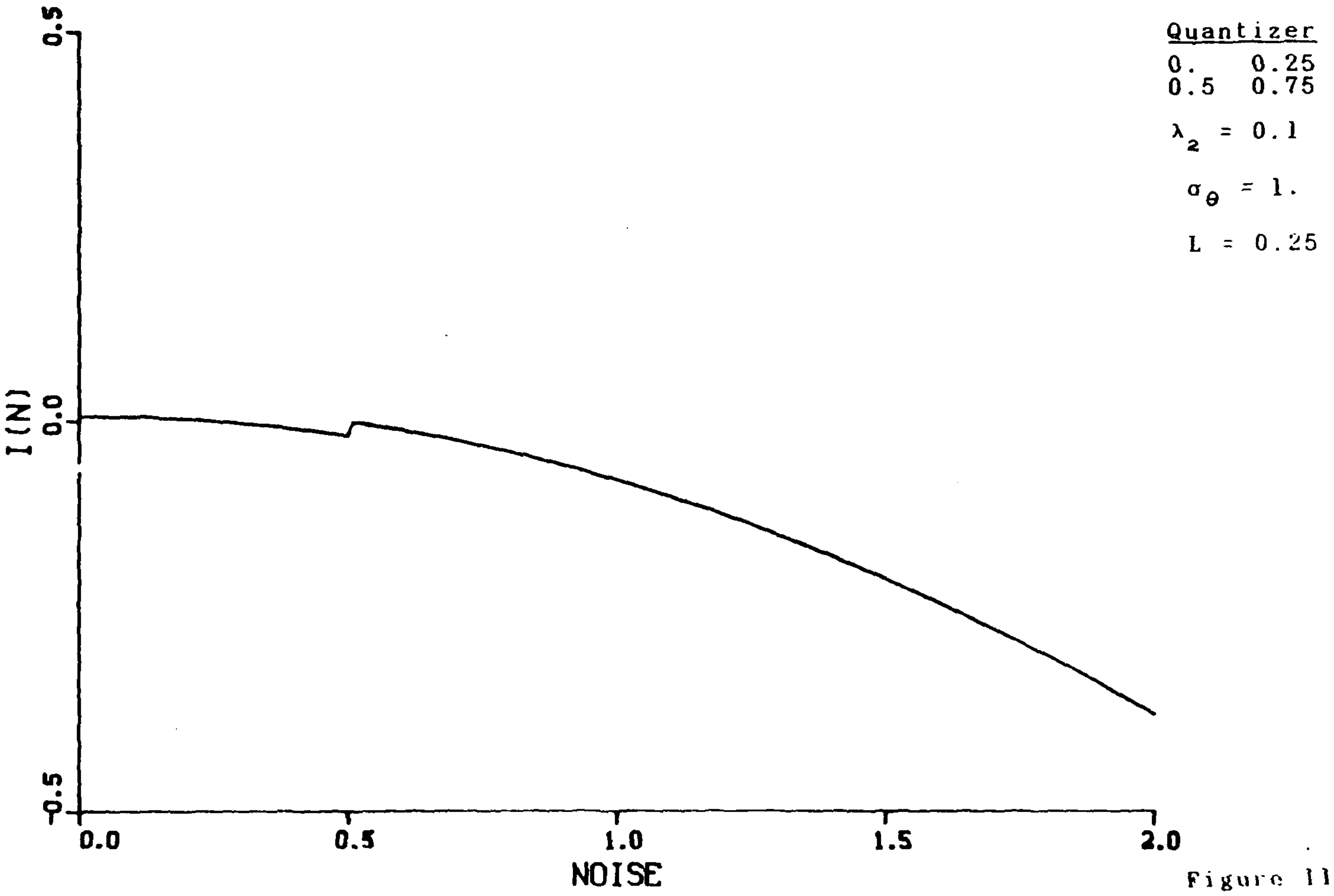


Figure 11

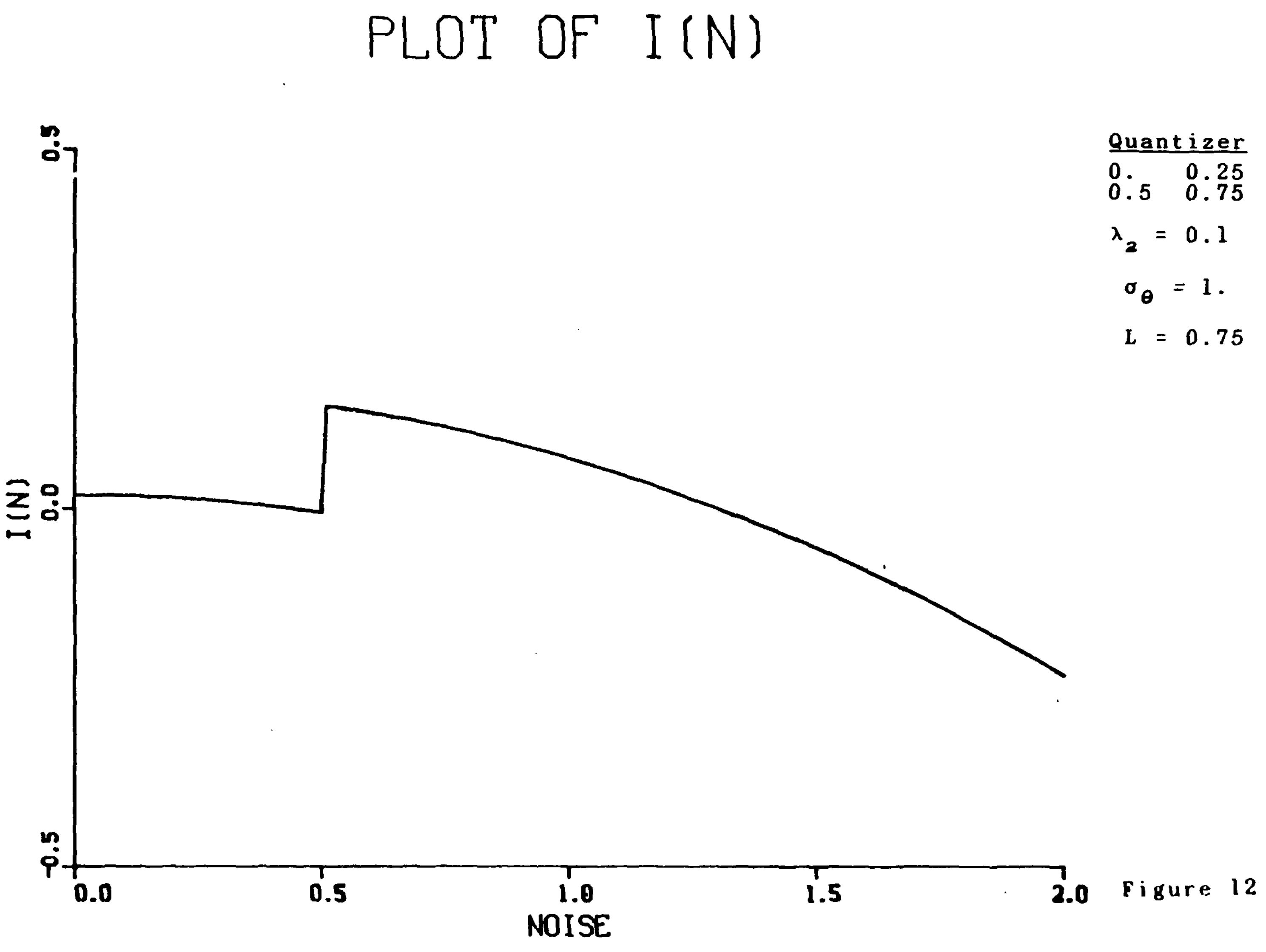

PLOT OF I(N)
Quantizer
0. 0.25
0.5 0.75
$\lambda_2 = 0.1$
$\sigma_\theta = 1.$
L = 0.75
0.5
0.0
-0.5
I(N)
0.0
0.5
1.0
1.5
2.0
NOISE

Figure 12

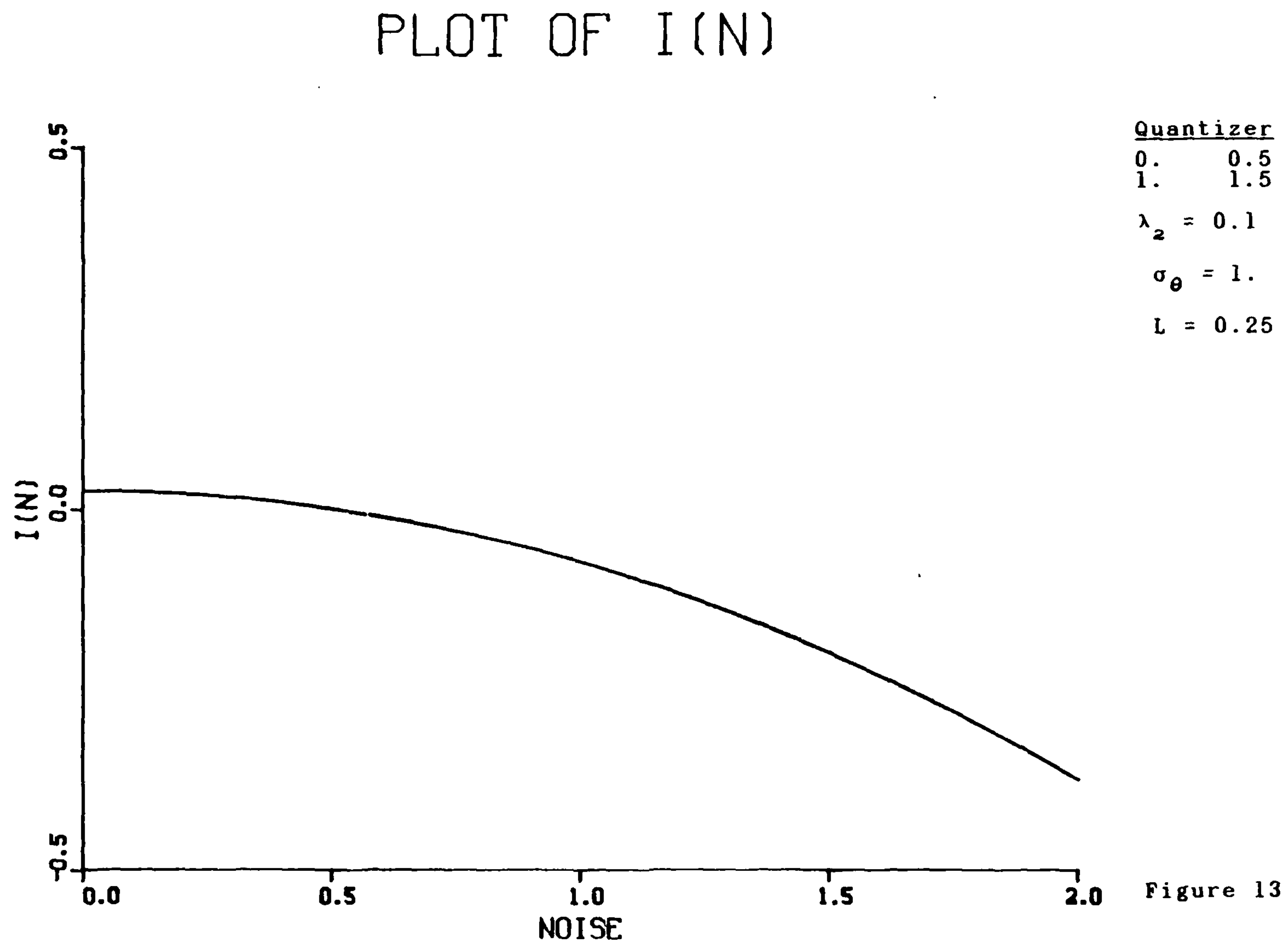

PLOT OF I(N)
0.5
0.0
-0.5
I(N)
0.0
0.5
1.0
1.5
2.0
NOISE
Quantizer
0. 0.5
1. 1.5
λ₂ = 0.1
σθ = 1.
L = 0.25

Figure 13

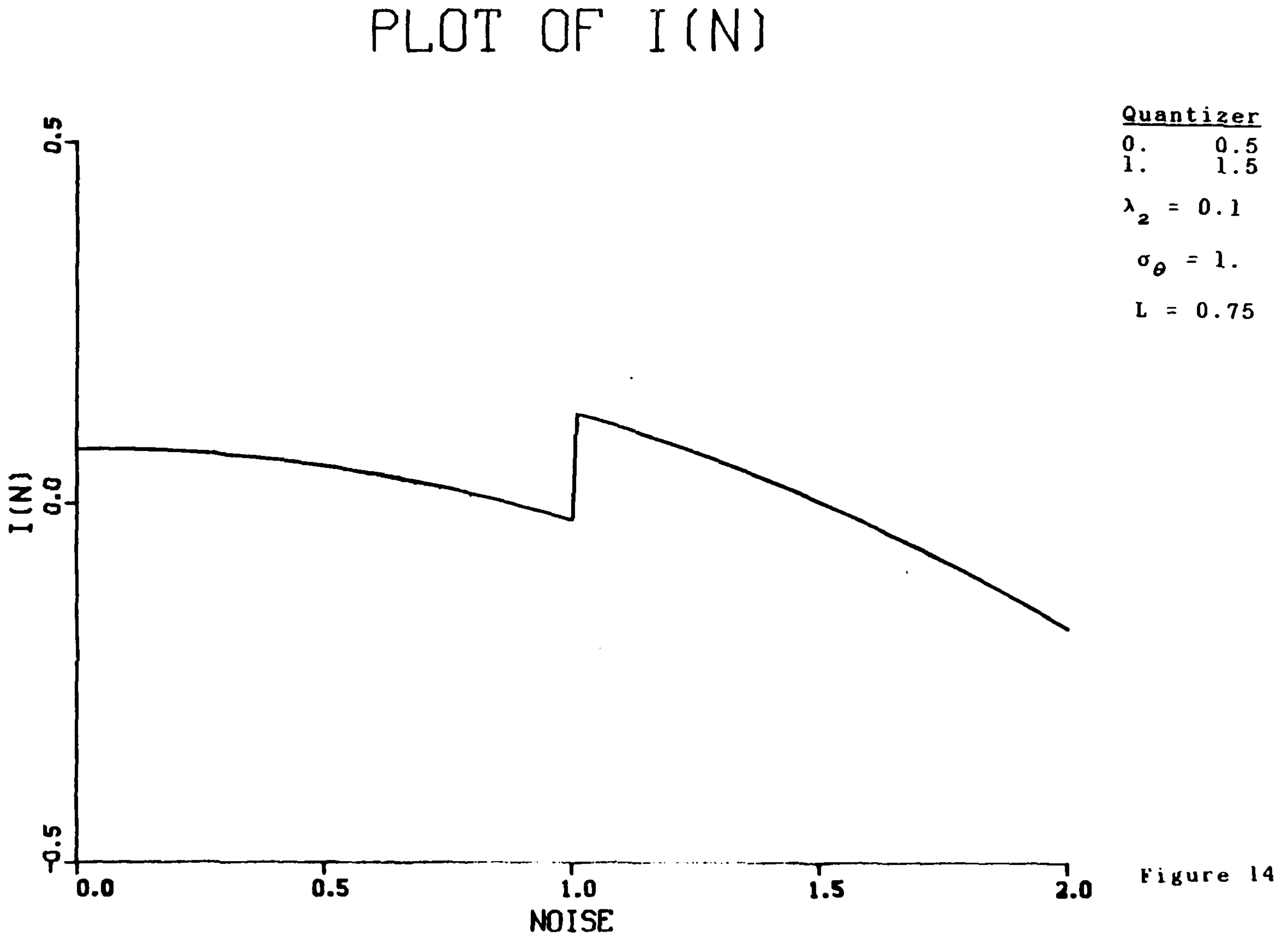

PLOT OF I(N)
Quantizer
0. 0.5
1. 1.5
$\lambda_2 = 0.1$
$\sigma_\theta = 1.$
L = 0.75
0.5
0.0
-0.5
I(N)
0.0
0.5
1.0
1.5
2.0
NOISE

Figure 14

$$\mathcal{R}^* = \min_{q \in \mathcal{Q}_N} \min_{\substack{\lambda_1, \lambda_2 \geq 0 \\ I(\eta) \leq 0}} (\lambda_1 + \lambda_2 c) \qquad (4.18)$$

This result is similar to that obtained for minimax quantization by Bath and VandeLinde for the generalized moment constrained class of quantizer input distributions. It differs primarily in the function $I(\eta)$ which incorporates the cost of estimating the source by the quantization mapping and the *a priori* knowledge of the clean source probability distribution function, $F_\theta(\theta)$.

The minimization of equation (4.13) over all nonnegative λ_1, λ_2, and nonpositive $I(\eta)$ implies the maximum of $I(\eta)$ is zero. As can be seen in equation (4.15), the role played by λ_1 in $I(\eta)$ is merely that of a bias term. This implies that the minimization of equation (4.13) can be determined by solving:

$$\lambda_1^*(\lambda_2) = \max_{\eta} \left[\int_{\Theta} C(\theta, q(x)) \, p_\theta(\theta) \, d\theta - \lambda_2 \rho(\eta) \right] \qquad (4.19)$$

and minimizing over all possible $\lambda_2 \geq 0$. Maximization of $I(\eta)$ with respect to λ_2 and $\lambda_1^*(\lambda_2)$ for a specific quantizer $q \in \mathcal{Q}_N$ will then yield the minimum λ_2 and the requisite λ_1 for $\max_{\lambda_1, \lambda_2 \geq 0} I(\eta) = 0$.

The minimax risk quantizer determination procedure, while quite numerically tedious, may be described by the following:

1) Select $\lambda_2 \geq 0$.
2) Determine the optimal λ_1 for the specified λ_2: $\lambda_1^*(\lambda_2)$.
3) Repeat step 2) over all $\lambda_2 \geq 0$ to determine the minimum of $(\lambda_1^*(\lambda_2) + \lambda_2 c)$. This is the maximum risk for the quantizer q: $\mathcal{R}^*(q)$.
4) Repeat steps 1)-3) over all quantizers $q \in \mathcal{Q}_N$ to determine the minimum of the maximum risks: $\mathcal{R}^* = \min_{q \in \mathcal{Q}_N} \mathcal{R}^*(q)$.
5) The minimax risk quantizer is that quantizer q^* which yields the minimax risk: $\mathcal{R}^*(q^*) = \mathcal{R}^*$.

4.2.3 Gaussian Source Simulation Example

Table 3 lists the results of the minimax risk quantization simulation compared to the Max-Lloyd and

TABLE 3

4-Level Quantizers

Gaussian Source + Gaussian Noise Example

(Squared-Error Cost)

$\mu_\theta = \mu_N = 0$

	Max-Lloyd		Minimum Risk		Minimax-Risk*		
S/N (dB)	x_i	y_i	x_i	y_i	x_i	y_i	c
8	0	0.4874	0	0.4774	0	0.49	0.16
	1.056	1.625	1.056	1.298	1.2	1.25	
4	0	0.5354	0	0.383	0	0.15	0.4
	1.1604	1.7854	1.1604	1.277	1.1	1.0	
0	0	0.6404	0	0.3202	0	0.13	1.0
	1.388	2.136	1.388	1.0678	1.4	1.3	

*Minimax-Risk quantizer calculated with: $L = 0.25$

$p_\theta(\theta) \sim N(0,1)$

stepsize $= \dfrac{\sqrt{c}}{25}$

minimum risk quantizers for a Gaussian source with the squared error cost function. The minimum risk quantizer assumes the noise to also be Gaussian. While the simulation results for the minimax risk quantizer appear somewhat ambiguous, there are several reasons which offer plausible explanation. First, the calculations involved in the Lagrange minimization are enormous and the precision in the calculation could easily influence the outcome of the calculated maximum risk for any given quantizer. Second, the precision of integration over the known density function can further influence the risk calculation. For the results presented here, the integration was approximated by a Riemann sum of 25 samples of the density function over $[\mu_\theta - 2\sigma_\theta, \mu_\theta + 2\sigma_\theta]$, which can further affect the risk. Third, the determination of the maximum of $I(n)$, over all possible N, due to computational constraints, must be limited to a sampled range. The range used here was $N \in [\pm 2\sqrt{c}]$, sampled in 50 equivalent intervals. As illustrated in the plots of $I(n)$, this may cause some error in determining the maximum if the samples miss the region of the discontinuity in the function. And finally, since these calculations must be performed over "all" N-level quantizers (not limited to levels appearing within specific ranges as in classical quantization), the step size imposed may effectively predetermine a suboptimal solution. The quantizer step size used here was the same as the step over the noise range.

It appears from the plots of $I(n)$ that the areas of maxima occur at or very near the quantizer thresholds. While it will not be shown here that this is the case, a simulation has been performed based on this assumption with the results appearing in tables 4 and 5. These results were determined by sampling $I(n)$ in the region surrounding each threshold, with greater accuracy imposed in all calculations than those in the previous simulation. Specifically, the quantizer step size was fixed at 0.01, the number of intervals in the Riemann sum was increased, and the accuracy of the Lagrange minimization was increased to 1×10^{-7}. The minimax risk quantizer with specific quantizer thresholds were found, and the minimum of these selected as the minimax risk quantizer.

Comparisons of the minimax risk quantizers to the Max-Lloyd and minimum risk quantizers (G.S.+G.N.) are shown at the bottom of tables 4 and 5. At 8 dB, the thresholds of all three are nearly identical, with the levels for the minimum risk quantizer inside (closer to the mean) those of the Max-Lloyd quantizer, and the

TABLE 4

Minimax Quantizer Determination
(N = 4, Squared-Error Cost, $p_\theta \sim N(0,1)$)
$\mu_N = 0$, L = 0.25, SNR = 8dB $\Rightarrow$ c = 0.1585

x_1	y_0	y_1	Minimax-Risk
0.90	0.41	1.23	0.09931
0.91	0.41	1.24	0.09854
0.92	0.41	1.24	0.09780
0.93	0.41	1.24	0.09708
0.94	0.41	1.24	0.09638
$\longrightarrow$ 0.95	0.42	1.24	0.09571 $\longleftarrow$
0.96	0.42	1.28	0.09743
0.97	0.42	1.31	0.09921
0.98	0.41	1.31	0.09875
0.99	0.41	1.31	0.09818
1.00	0.41	1.31	0.09762
1.01	0.41	1.32	0.09709
1.02	0.41	1.32	0.09657
1.03	0.41	1.32	0.09606
1.04	0.41	1.36	0.09807
1.05	0.41	1.39	0.09973

Max-Lloyd		Minimum Risk*		Minimax Risk	
x_i	y_i	x_i	y_i	x_i	y_i
0	0.4874	0	0.4774	0	0.42
1.056	1.625	1.056	1.298	0.95	1.24

*Minimum Risk for Gaussian Sources.

TABLE 5

Minimax Quantizer Determination
(N = 4, Squared-Error Cost, $p_\theta \sim N(0,1)$)

$\mu_N = 0$, $L = 0.25$, SNR = 4dB $\Rightarrow$ $c = 0.398$

x_1	y_0	y_1	Minimax-Risk
1.27	0.41	1.51	0.12232
1.28	0.41	1.53	0.12372
1.29	0.41	1.56	0.12500
1.30	0.41	1.56	0.12447
1.31	0.41	1.56	0.12395
1.32	0.41	1.57	0.12343
1.33	0.41	1.57	0.12294
1.34	0.41	1.57	0.12254
$\longrightarrow$ 1.35	0.41	1.57	0.12198 $\longleftarrow$
1.36	0.41	1.58	0.12348
1.37	0.41	1.60	0.12492
1.38	0.41	1.60	0.12450
1.39	0.41	1.60	0.12408
1.40	0.41	1.60	0.12367
1.41	0.41	1.61	0.12326

Max-Lloyd		Minimum Risk*		Minimax Risk	
x_i	y_i	x_i	y_i	x_i	y_i
0	0.5354	0	0.3830	0	0.41
1.160	1.785	1.160	1.277	1.35	1.57

*Minimum Risk for Gaussian Sources.

minimax risk quantizer levels just slightly inside those of the minimum risk quantizer. At 4 dB, the outer threshold of the minimax risk quantizer has migrated well outside of the other two quantizers, which may be from the Max-Lloyd and minimum risk quantizers assumption of Gaussian densities, while the minimax risk quantizer only considers constrained noise power. The levels of the minimax risk quantizer are now between those of the Max-Lloyd and minimum risk quantizers. This may be interpreted from the standpoint of the assumptions also. The minimum risk quantizer assumes both the source and noise are Gaussian, so it arrives at closely compacted estimates for the levels, while the minimax risk quantizer only assumes the source is Gaussian for this example, and the noise power constrained -- so its estimates are outside those of the minimum risk quantizer (for this example). Compared to the Max-Lloyd quantizer, the minimax risk quantizer is "aware" of the corrupting influence of the noise, whereas the Max-Lloyd is not and the estimates for the quantizer levels of the minimax risk quantizer are then placed within those of the Max-Lloyd quantizer.

4.3 Minimax Risk Quantizer for Known Noise Statistics

The minimax risk theory just applied to the problem of quantizing an i.i.d. source, with known probability distribution, corrupted by a sequence of independent additive noise random variables each of uncertain distribution, is easily modified for the converse problem. If the noise sequence is i.i.d. with known c.p.d.f. $F_{\mathcal{N}}$, but the source sequence $\{\theta_k\}$ is only known to be distributed with the c.p.d.f. sequence $\{F_{\theta_k}\}$ from the set C, then the risk may be written:

$$R(q,F_\theta,F_{\mathcal{N}}) = \int_\theta \int_{\mathcal{N}} C(\theta,q(x))\ p_{\mathcal{N}}(n)\ dn\ dF_\theta \qquad (4.20)$$

The minimax risk quantizer development for this problem parallels the development for the previous case. A similar solution is described as the minimax risk quantizer q^* which yields the minimum maximal risk, $R^*(q)$, where:

$$R^*(q) = \min_{\substack{\lambda_1,\lambda_2\geq 0 \\ I(\theta)\leq 0}} (\lambda_1 + \lambda_2 c) \qquad (4.21)$$

and

$$I(\theta) = \int_{\mathcal{N}} C(\theta - q(x))\ p_{\mathcal{N}}(n)\ dn - \lambda_1 - \lambda_2\rho(\theta) \qquad (4.22)$$

The same restrictions of the previous section apply to the distortion and constraint functions, $d(\theta, q(x))$ and $\rho(\theta)$. Likewise,

$$\lambda_1^*(\lambda_2) = \max_{\theta} \left[\int_{N} C(\theta, q(x)) \; p_N(n) \; dn - \lambda_2 \rho(\theta) \right] \quad (4.23)$$

and the minimax risk quantizer q^* is specified by a procedure analogous to that of the previous section.

Figures 15 through 18 show the effects due to the constraint on the noise power and the distortion bound L on the function $I(\theta)$. These appear roughly similar to $I(s)$ in [4], differing through the knowledge of the noise for $I(\theta)$.

Note that in the noiseless case, this risk definition coincides precisely with the distortion function of an unknown input sequence used by Bath and VandeLinde. The minimax risk quantizer then is identical to the minimax quantizer developed there for the classical quantizer distortion measure.

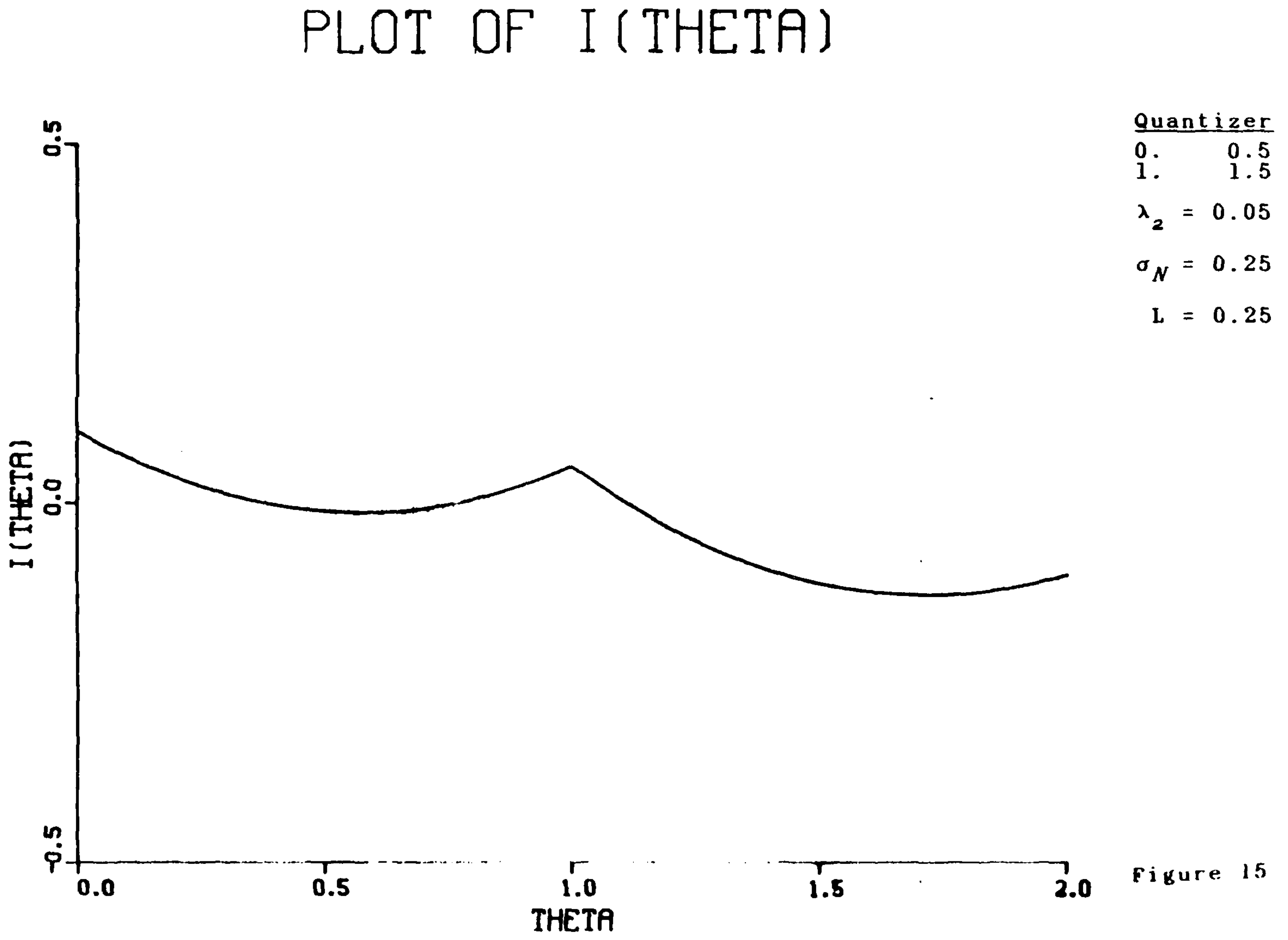

PLOT OF I(THETA)
Quantizer
0. 0.5
1. 1.5
λ2 = 0.05
σN = 0.25
L = 0.25
0.5
0.0
-0.5
I(THETA)
0.0
0.5
1.0
1.5
2.0
THETA

Figure 15

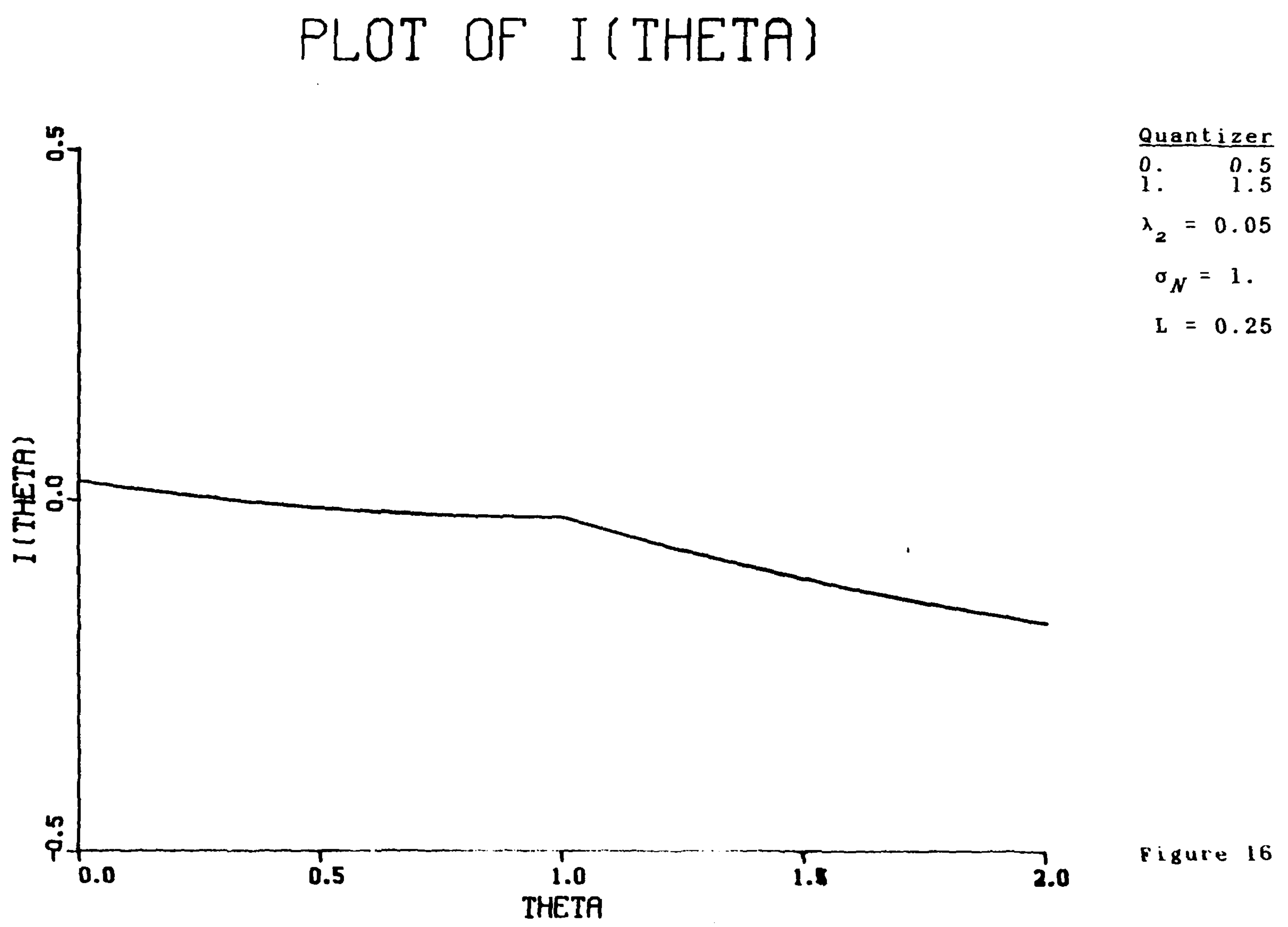

PLOT OF I(THETA)
0.5
0.0
-0.5
I(THETA)
0.0
0.5
1.0
1.5
2.0
THETA
Quantizer
0. 0.5
1. 1.5
$\lambda_2 = 0.05$
$\sigma_N = 1.$
L = 0.25

Figure 16

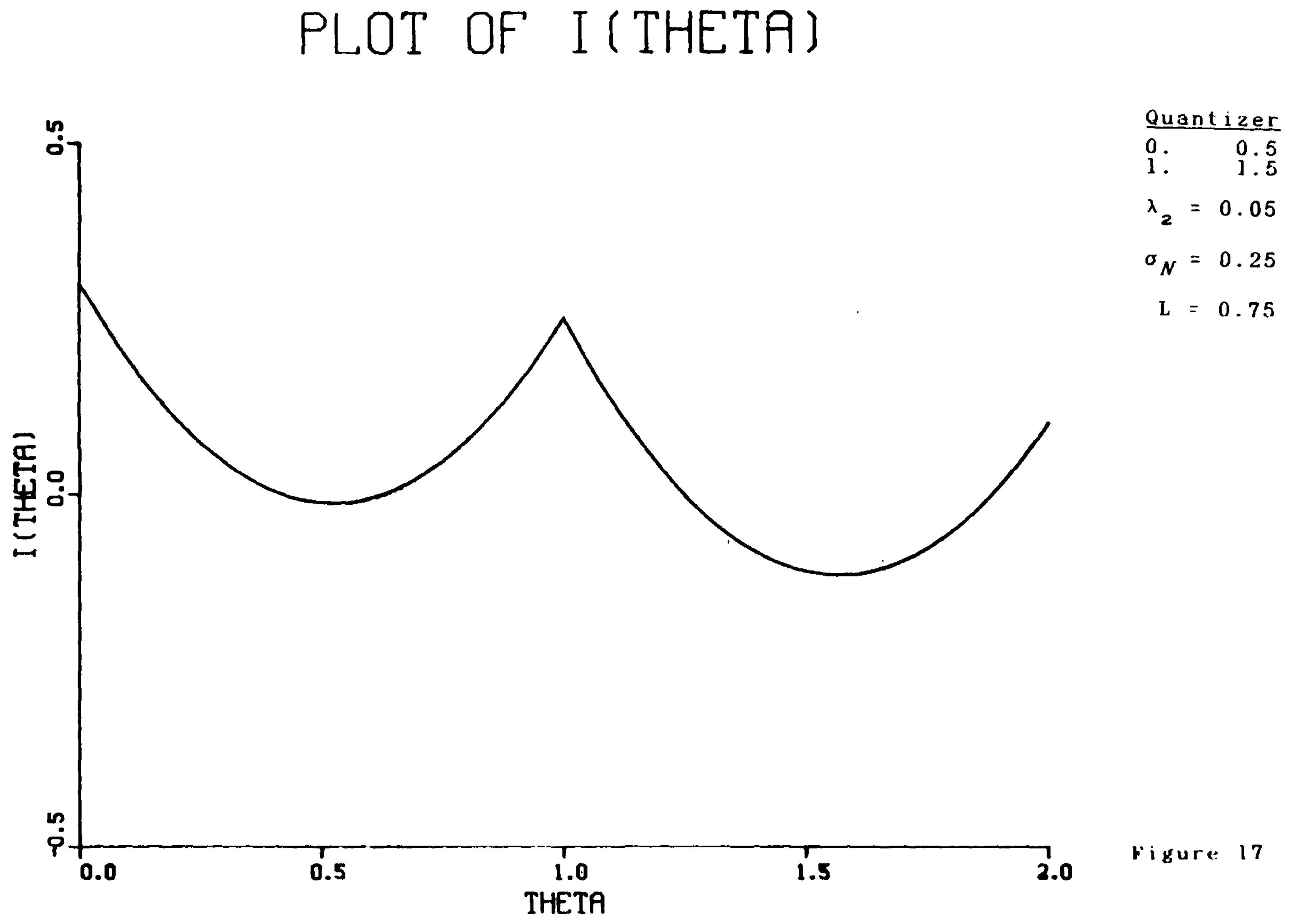

PLOT OF I(THETA)
Quantizer
0. 0.5
1. 1.5
$\lambda_2 = 0.05$
$\sigma_N = 0.25$
L = 0.75
0.5
0.0
-0.5
I(THETA)
0.0
0.5
1.0
1.5
2.0
THETA

Figure 17

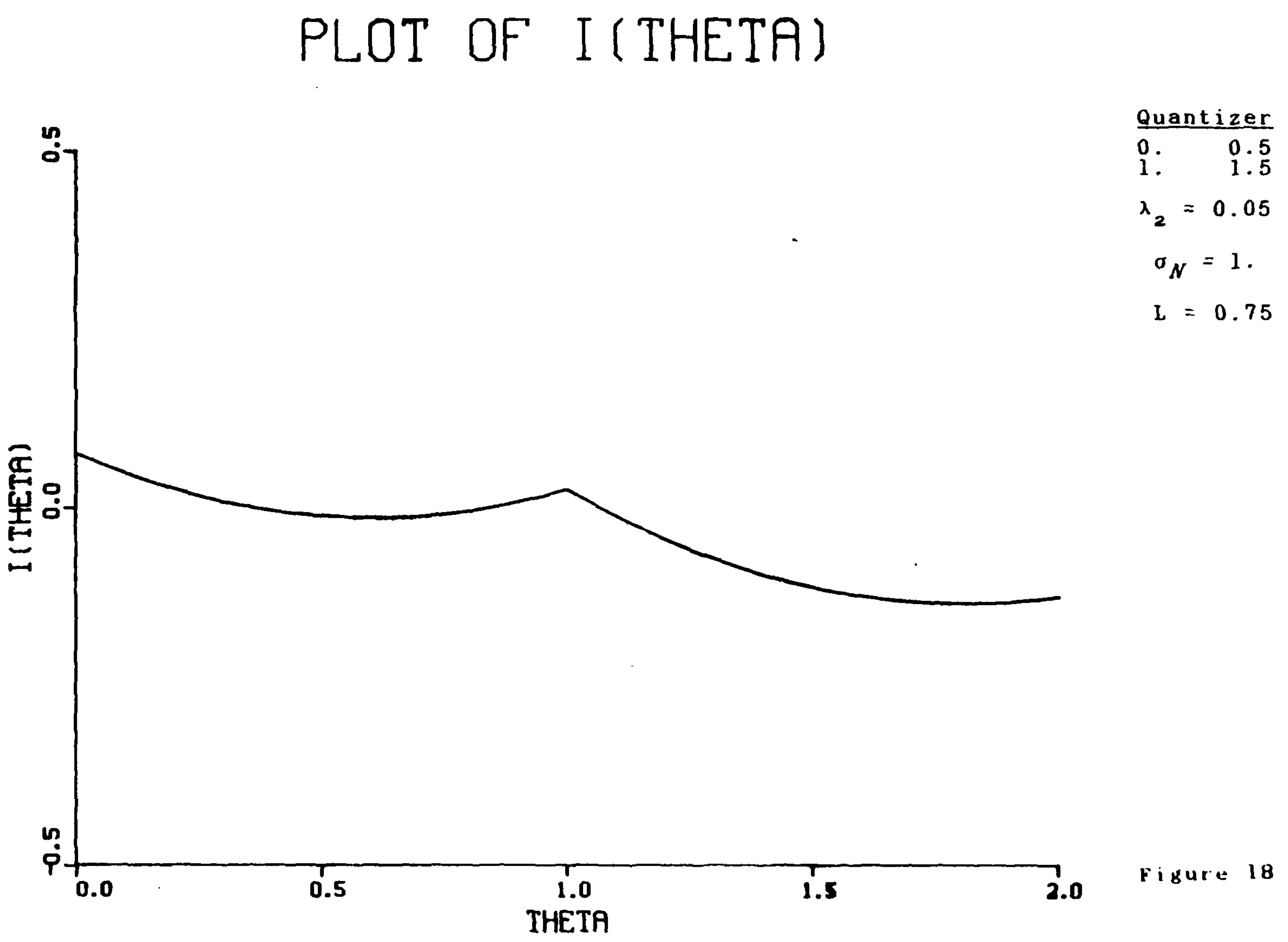

PLOT OF I(THETA)
I(THETA)
0.5
0.0
-0.5
0.0
0.5
1.0
1.5
2.0
THETA
Quantizer
0. 0.5
1. 1.5
$\lambda_2 = 0.05$
$\sigma_N = 1.$
L = 0.75

Figure 18

Section V

RESULTS AND CONCLUSIONS

A quantizer which compensates for noise should be quite useful in signal processing systems. The theory developed for the quantizer design is unique and presents a new way of thinking about quantization by modifying the normal definition of the quantizer as an input estimator to a general estimation device. The definition of a risk associated with an estimation of the source by the quantizer is introduced to provide a means for determining the specific quantizer mapping. Achieving a minimum risk criterion is the goal of whichever estimation technique is employed.

5.1 Minimum Risk Quantization

For the case where the source to be estimated is the quantizer input, the extended quantizer definition reduces to the traditional definition [16], and the development of the minimum risk quantizer parallels that of the optimal distortion development. Necessary conditions for minimum risk quantization have been derived and specific expressions for the squared error, absolute error, and uniform error cost functions have been developed. For the specific example of Gaussian source plus Gaussian noise, quantizers were derived with these three cost functions, and tables generated for the squared error and absolute error cost functions at various signal-to-noise rates.

Quantizer performance is normally measured with respect to the quantizer distortion (based on the input). For the class of risk quantizers developed here, this would have no meaning since the object of risk quantization is that of original source estimation and not the noisy input. The quantizers must be compared then to the accuracy of estimation of the source, which is provided by the risk function itself.

5.2 Images

While the risk surfaces show the respective performance in terms of signal variance to noise variance ratio (SNR) for various noise statistics, as a more graphic demonstration of the quantizer performances, sample images were quantized and compared. Some of these images appear in appendix D and illustrate the basic properties of the minimum risk quantizer.

The images presented show the effects of source quantization with Max-Lloyd and minimum risk quantizers for noise-corrupted sources. The source considered, a picure element (pixel) of a digital image, represents the light intensity of a minute area of a monochromatic picture. The source is assumed Gaussian and the squared error cost function is used to design the quantizers. The statistics for the source were estimated by the statistics of a sampled window about the face of the person in the image.

While it is realized that the assumption of independent Gaussian statistics for the image is a crude approximation to the actual situation, the images demonstrate the superiority of the minimum risk quantization to that of Max-Lloyd and uniform quantization. They further illustrate some of the basic properties of all the quantizers. Note particularly how the Max-Lloyd quantizer follows the observation mean when the noise is caused to offset the mean of the image, whereas the risk quantizer compensates accordingly. The original image and the noisy images are repre sented by 256-level gray scale, while all remaining images were quantized with 4-level quantizers. The images (appendix D) are presented in the following format:

Original Image Source 256 Levels	Max-Lloyd Original 4 Level	Max-Lloyd Orig+Noise 4 Level
Noisy Source 256 Levels	Uniform 4 Level	Min. Risk 4 Level

where "Original" refers to a Max-Lloyd quantizer designed solely for the clean source image and "Orig+Noise" refers to one with is designed for the observation image of source + noise.

Sample images were also examined for the absolute error cost function, and the results were similar to those for the squared error cost function. A comparison of the subjective image quality between the two cost measures did not yield a significant difference between them, so sample photos for the absolute error cost function are not included in this dissertation.

5.3 <u>Minimax-Risk Quantization</u>

The risk functions examined for the minimax risk quantization indicate the ability to quantize a source when the source statistics and probability density function are well-defined, but the noise probability density function is known to belong only to a general class of density functions (the generalized moment constrained class). The process has also been shown reversible and may be applied to the problem when well-defined noise statistics and distribution (such as specific characteristics of a sensor, or camera effects, are imposed on the quantization process, but the source distribution is primarily unknown.

Both of these situations are based upon the risk function definition, which as the noise power decreases, converge to well-known classical results. Specifically, for the known source corrupted by unknown noise, as the noise power approaches zero, the risk function becomes precisely that of the quantizer distortion of Max [16]. Likewise, when the noise statistics are well-defined, but source is unknown, as the problem approaches the noiseless case, the risk function becomes the quantizer distortion used by Bath [4] for the minimax quantizer distortion.

5.4 Conclusions

The theory and results presented here, using a new risk theory approach to quantizer design, demonstrate significant improvement over classical designs for noisy sources. For independent additive noise, the quantization problem has been formulated and solved, both for well-known noise distributions and uncertain distributions. Also, when the source statistics are uncertain but the noise is well-defined, a minimax risk quantization may be achieved.

The risk theory approach suggests quantization schemes for other problems, such as correlated samples or multiplicative noise, may now be developed. Also, the approach appears extendable to the problem of multidimensional quantization, and suggests possible advancements in the area. Further, the same techniques employed for the quantizer design development shows the promise of the risk theory approach to other applications as well as that of quantization, such as neural network theory.

Bibliography

BIBLIOGRAPHY

[1] Aazhang, B., and Poor, H.V., "On Optimum and Nearly Optimum Data Quantization for Signal Detection," IEEE Trans. on Comm., Vol. COM-32, No. 7, July 1984, pp. 745-751.

[2] Abaya, E., and Wise, G.L., "Some Notes on Optimal Quantization," ICC '83, pp. 30.7.1- 30.7.5.

[3] Bath, W.G., "Robust Quantization of Independent and Dependent Data," Ph.D. dissertation, Dept. of Elec. Eng., Johns Hopkins Univ., Baltimore, MD, 1980.

[4] Bath, W.G., and VandeLinde, V.D., "Robust Memoryless Quantization for Minimum Signal Distortion," IEEE Trans. on Info. Theory, Vol. IT-28, No. 2, March 1982, pp. 296-306.

[5] Berger, T., Rate Distortion Theory, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1971.

[6] Farvardin, N., and Vaishampayan, V., "Optimal Quantizer Design for Noisy Channels: An Approach to Combined Source-Channel Coding," Submitted to IEEE Trans. on Info. Theory.

[7] Fisher, T.R., and Gibson, J.D., "Estimation and Optimum Source Coding of Noisy Sources," ICC '86, pp. 59.1.1-59.1.5.

[8] Gallager, R.G., Information Theory and Reliable Communication, John Wiley and and Sons, Inc., 1968.

[9] Gersho, A., "On the Structure of Vector Quantizers," IEEE Trans. on Info. Theory, Vol. IT-28, No. 2, March 1982, pp. 157-166.

[10] Gersho, A., "Principles of Quantization," IEEE Trans. on Circuits and Systems, Vol. CAS-25, Number 7, July 1978, pp. 427-436.

[11] Gray, R.M., "Mutual Information Rate, Distortion, and Quantization in Metric Spaces," IEEE Trans. on Info. Theory, Vol. IT-26, No. 4, July 1980, pp. 412-422.

[12] Kanaya, F., and Oishi, S., "Rate Risk Function Theory," IEEE Int. Symposium on Info. Theory, 1985.

[13] Kazakos, D., "New Results on Robust Quantization," IEEE Trans. on Comm., Vol. COM-31, No. 8, Aug. 1983, pp. 965-974.

[14] Lloyd, S.P., "Least Squares Quantization in PCM," IEEE Trans. on Info. Theory, Vol. IT-28, Number 2, March 1982, pp. 129-137.

[15] Luenberger, D.G., Optimization by Vector Space Methods, John Wiley and Sons, Inc., New York, 1969.

[16] Max, J. "Quantizing for Minimum Distortion," IRE Trans. on Info. Theory, Vol. IT-6, March 1960, pp. 7-12.

[17] Melsa, J.L., and Cohn, D.L., Decision and Estimation Theory, McGraw-Hill, Inc., New York, 1978.

[18] Mood, A.M., Graybill F.A., and Boes, D.C., Introduction to the Theory of Statistics, McGraw-Hill Book Co., New York, 1974.

[19] Morris, J.M., and VandeLinde, V.D., "Robust Quantization of Discrete-Time Signals with Independent Samples," IEEE Trans. on Comm., Vol. COM-22, No. 12, Dec. 1974, pp. 1897-1902.

[20] Royden, H.L., Real Analysis, Macmillan Publishing Co., Inc., 1968.

[21] Sharma, D.K., "Design of Absolutely Optimal Quantizers for Wide Class of Distortion Measures," IEEE Trans. on Info. Theory, Vol. IT-24, No. 6, Nov. 1978, pp. 693-701.

[22] Van Trees, H.L., Detection, Estimation and Modulation Theory, Part I, John Wiley & Sons, Inc., New York, 1968.

[23] Widder, D.V., Advanced Calculus, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1964.

[24] Wolf, J.K., and Ziv, J., "Transmission of Noisy Information to a Noisy Receiver with Minimum Distortion," IEEE Trans. on Info. Theory, Vol. IT-16, No. 4, July 1970, pp. 406-411.

APPENDIX A

APPENDIX A

The threshold condition for the absolute error cost function was given in (3.16) as:

$$\int_{-\infty}^{\infty} \theta \, p_{\theta|X}(\theta|x_i) \, d\theta = \frac{y_i+y_{i-1}}{2}$$
$$+ \int_{-\infty}^{y_{i-1}} (\theta-y_{i-1}) \, p_{\theta|X}(\theta|x_i) \, d\theta + \int_{y_i}^{\infty} (\theta-y_i) \, p_{\theta|X}(\theta|x_i) \, d\theta \qquad (A.1)$$

where $\{y_i\}$, $i=1,\ldots,N$, are ordered $y_{i-1} < y_i$. Define

$$\mu_i = \int_{-\infty}^{\infty} \theta \, p_{\theta|X}(\theta|x_i) \, d\theta \qquad (A.2)$$

for all i. Now if $p_{\theta|X}(\theta|x_i)$ is symmetric about μ_i, then

$$\int_{-\infty}^{\infty} \theta \, p_{\theta|X}(\theta|x_i) \, d\theta = \frac{y_i+y_{i-1}}{2}, \qquad \forall i = 1,\cdots,N \qquad (A.3)$$

Proof

Case 1) $\mu_i \le \frac{y_i+y_{i-1}}{2}$

By symmetry:

$$\int_{y_i}^{\infty} (\theta - y_i)\; p_{\theta|X}(\theta|x_i)\; d\theta$$

$$= -\int_{-\infty}^{2\mu_i - y_i} (\theta - (2\mu_i - y_i))\; p_{\theta|X}(\theta|x_i)\; d\theta \qquad (A.4)$$

$$\mu_i = \frac{y_i + y_{i-1}}{2} + \int_{-\infty}^{2\mu_i - y_i} (2\mu_i - y_i - y_{i-1})\; p_{\theta|X}(\theta|x_i)\; d\theta$$

$$+ \int_{2\mu_i - y_i}^{y_{i-1}} (\theta - y_{i-1})\; p_{\theta|X}(\theta|x_i)\; d\theta \qquad (A.5)$$

It is desired to determine all possible μ_i such that (A.5) exists. At $\mu_i = \frac{y_i + y_{i-1}}{2}$, the last two terms on the R.H.S. are zero and the expression is satisfied. Now consider (31) as a condition of:

$$f(\mu_i) = g(\mu_i) \qquad (A.6)$$

where $f(\mu_i) = \mu_i$ (L.H.S. of (A.5)) and $g(\mu_i)$ = R.H.S. of (A.5). It is necessary to determine all μ_i such that (A.6) exists. It has been shown that $f(\mu_1) = g(\mu_1)$ for $\mu_1 = \frac{y_i + y_{i-1}}{2}$, so it is necessary to examine (A.6) for all $\mu_i < \mu_1$. Taking the derivative of each function w.r.t. μ_i:

$$f'(\mu_i) = 1$$

$$g'(\mu_i) = \int_{-\infty}^{2\mu_i - y_i} 2\, p_{\theta|X}(\theta|x_i)\, d\theta \qquad (A.7)$$

by Leibnitz' rule. For $\mu_i < \mu_1$ and $y_{i-1} < y_i \Rightarrow \mu_i < y_i$ $\Rightarrow \mu_i < y_i \Rightarrow 2\mu_i - y_i < \mu_i$. In this case $g'(\mu_i)$ is always less than 1, approaching it as $\mu_i \to \mu_1$. Since it has been shown that the two functions intersect at μ_1, and their first derivatives are never equal at any other μ_i (for the region of interest) this implies the intersection at μ_1 is unique. See figure 19.

Case 2) $\mu_i \geq \frac{y_i + y_{i-1}}{2}$

The argument parallels that for case 1 with:

$$\int_{-\infty}^{y_{i-1}} (\theta - y_{i-1})\, p_{\theta|X}(\theta|x_i)\, d\theta$$
$$= -\int_{2\mu_i - y_{i-1}}^{\infty} (\theta - (2\mu_i - y_{i-1}))\, p_{\theta|X}(\theta|x_i)\, d\theta \qquad (A.8)$$

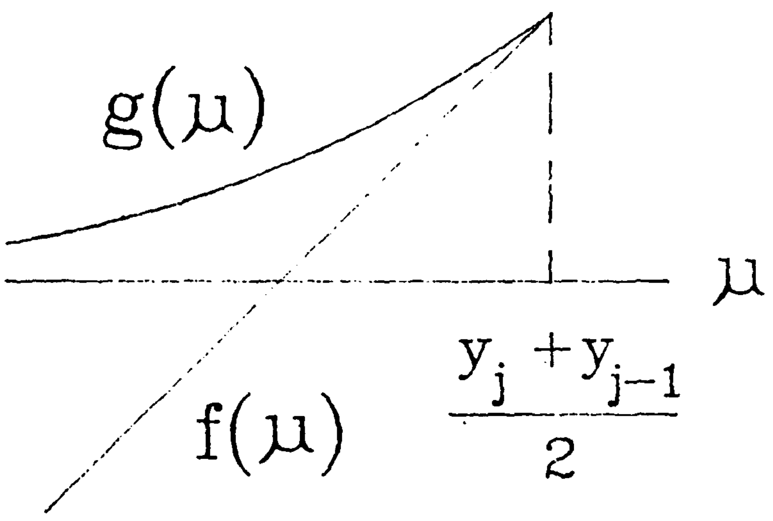


Figure 19

APPENDIX B

APPENDIX B

The threshold condition for the uniform cost function was:

$$\int_{y_{i-1}-\Delta/2}^{y_{i-1}+\Delta/2} p_{\theta|X}(\theta|x_i)\, d\theta = \int_{y_i-\Delta/2}^{y_i+\Delta/2} p_{\theta|X}(\theta|x_i)\, d\theta \qquad \text{(B.1)}$$

Now if $p_{\theta|X}(\theta|x_i)$ is continuous, symmetric about mean μ_i, and monotonic decreasing for $\theta \geq \mu_i$, it can be shown that it is sufficient to say:

$$\mu_i = \int_{-\infty}^{\infty} \theta\, p_{\theta|X}(\theta|x_i)\, d\theta = \frac{y_i+y_{i-1}}{2}, \quad \forall i = 1,\cdots,N \qquad \text{(B.2)}$$

See figure 20. By inspection, $\forall\ \Delta\ \nexists\ y_1, y_2$ where $\mu_i < y_1 < y_2$, $\ni$ (B.1) holds. Likewise, $\nexists\ y_1, y_2$ where $y_1 < y_2 < \mu_i$, for which (B.1) holds. And it easy to see that for fixed Δ and $y_1 < \mu_i < y_2$, (B.1) holds only for $\mu_i = \frac{y_1+y_2}{2}$. A counter-example is given in figure 21 of a nondecreasing, but symmetric $p_{\theta|X}(\theta|x_i)$.

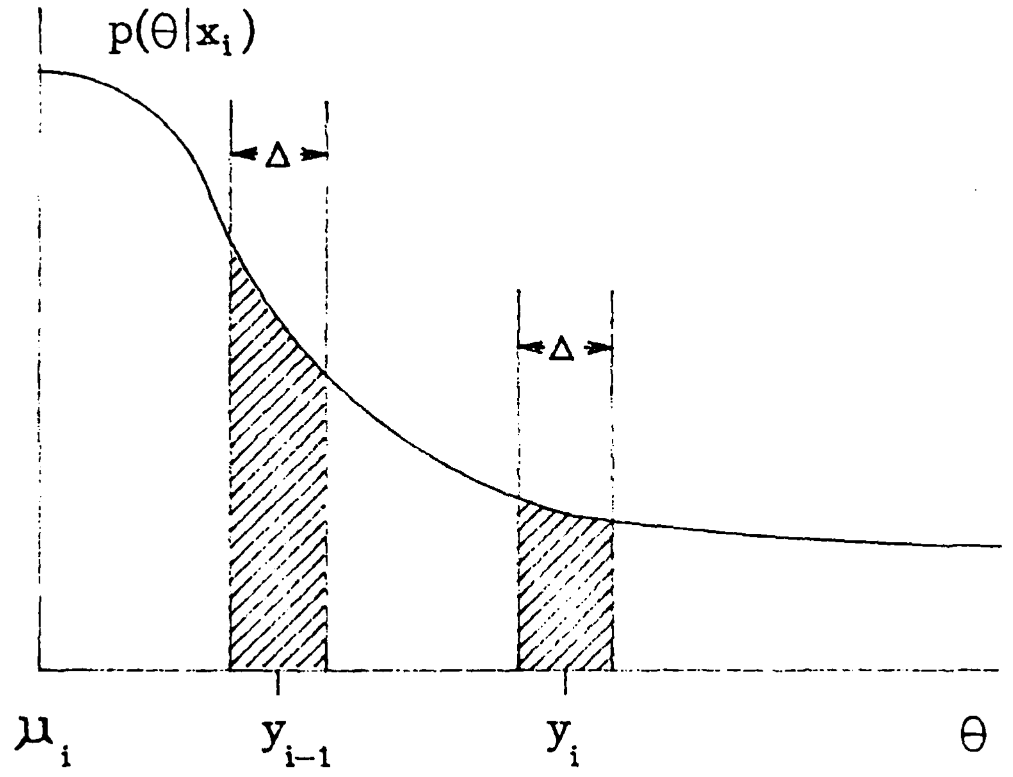


Figure 20

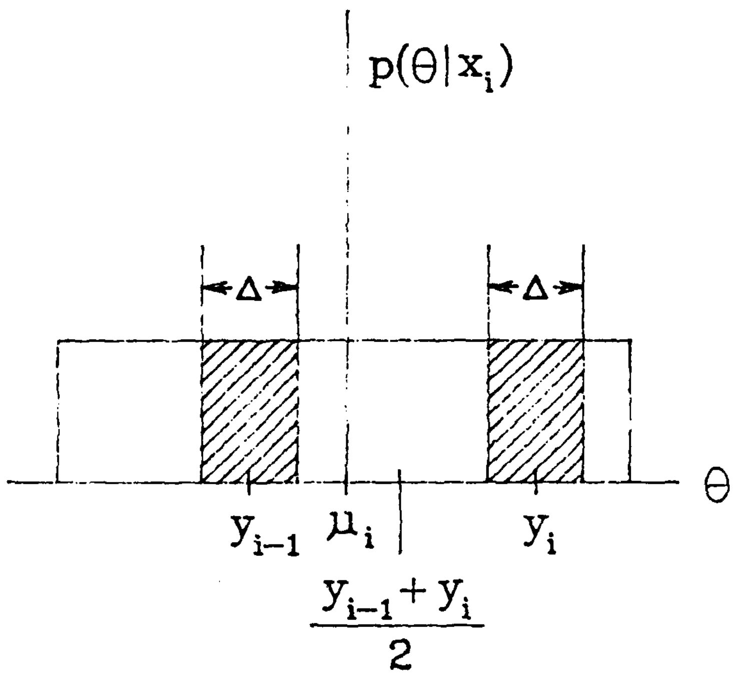


Figure 21

APPENDIX C

APPENDIX C

The following is the mathematical derivation used by the Gaussian source corrupted by Gaussian noise example of chapter III for the minimum-risk quantizer. If the source $\theta \sim N(\mu_\theta, \sigma_\theta)$ and noise $N \sim N(\mu_N, \sigma_N)$ are independent,

$$X = \theta + N \quad \Rightarrow \quad p_X(x) = p_\theta(x) \otimes p_N(x) = \int_{-\infty}^{\infty} p_N(x-\theta)\; p_\theta(\theta)\; d\theta \tag{C.1}$$

Since the sum of two independent Gaussian random variables is Gaussian, then $p_X(x) \sim N(\mu_\theta + \mu_N, \sqrt{\sigma_\theta^2 + \sigma_N^2}\,)$.

C.1. Squared Error Cost

For this example and the squared error cost function, equation (3.14) becomes:

$$y_i = \frac{\displaystyle\int_{x_i}^{x_{i+1}} \left[\int_{-\infty}^{\infty} \theta\; p_N(x-\theta)\; p_\theta(\theta)\; d\theta \right] dx}{\displaystyle\int_{x_i}^{x_{i+1}} p_X(x)\; dx}, \tag{C.2}$$

$$\forall i = 1, \cdots, N$$

Define:

$$g(x) = \int_{-\infty}^{\infty} \theta \; p_N(x-\theta) \; p_\theta(\theta) \; d\theta \qquad (C.3)$$

$$g(x) = \int_{-\infty}^{\infty} \frac{\theta}{2\pi\sigma_\theta\sigma_N} \exp\left\{ -\frac{1}{2}\left[\frac{\sigma_\theta^2(x-\theta-\mu_N)^2 + \sigma_N^2(\theta-\mu_\theta)^2}{\sigma_\theta^2 \; \sigma_N^2} \right]\right\} d\theta$$

Expanding:

$$\sigma_\theta^2(x-\theta-\mu_N)^2 + \sigma_N^2(\theta-\mu_\theta)^2 = (\sigma_\theta^2+\sigma_N^2)\,x^2 + (-2\sigma_\theta^2 x + 2\sigma_\theta^2\mu_N - 2\sigma_N^2\mu_\theta)\,x$$

$$+ \sigma_\theta^2 x^2 - 2\sigma_\theta^2\mu_N x + \sigma_\theta^2\mu_N{}^2 + \sigma_N^2\mu_\theta^2$$

$$= (\sigma_\theta^2+\sigma_N^2)\left[\left[\theta - \frac{\sigma_\theta^2 x - \sigma_\theta^2\mu_N + \sigma_N^2\mu_\theta}{\sigma_\theta^2+\sigma_N^2} \right]^2 - \left[\frac{-\sigma_\theta^2 x + \sigma_\theta^2\mu_N - \sigma_N^2\mu_\theta}{\sigma_\theta^2+\sigma_N^2} \right]^2 + \frac{\sigma_\theta^2 x^2 - 2\sigma_\theta^2\mu_N x + \sigma_\theta^2\mu_N^2 + \sigma_N^2\mu_\theta^2}{\sigma_\theta^2+\sigma_N^2} \right]$$

Define:

$$C_1 = \frac{\sigma_\theta^2+\sigma_N^2}{\sigma_\theta^2\sigma_N^2}, \qquad C_2 = \frac{\sigma_\theta^2 x - \sigma_\theta^2\mu_N + \sigma_N^2\mu_\theta}{\sigma_\theta^2+\sigma_N^2},$$

$$C_3 = -(C_2)^2 + \frac{\sigma_\theta^2 x^2 - 2\sigma_\theta^2\mu_N x + \sigma_\theta^2\mu_N{}^2 + \sigma_N^2\mu_\theta{}^2}{\sigma_\theta^2+\sigma_N^2}$$

Then $g(x)$ becomes:

$$g(x) = \int_{-\infty}^{\infty} \frac{\theta}{2\pi\sigma_\theta\sigma_N} \exp\left\{ -\frac{1}{2} C_1 \left[\left[x - C_2\right]^2 + C_3 \right\} d\theta \qquad (C.4)$$

$$g(x) = \frac{\exp\left\{\frac{-C_1C_3}{2}\right\}}{\sqrt{2\pi}\,\sqrt{\sigma_\theta^2+\sigma_N^2}} \int_{-\infty}^{\infty} \left[\frac{\theta}{\sqrt{2\pi}\,\sqrt{\sigma_\theta^2\sigma_N^2 \,/\, \sigma_\theta^2+\sigma_N^2}} \cdot \exp\left\{ \frac{-(\theta-C_2)^2}{2(\sigma_\theta^2\sigma_N^2 \,/\, \sigma_\theta^2+\sigma_N^2)} \right\} d\theta \right. \tag{C.5}$$

$$g(x) = \frac{C_2 \exp\left\{\frac{-C_1C_3}{2}\right\}}{\sqrt{2\pi}\,\sqrt{\sigma_\theta^2+\sigma_N^2}} = \frac{K_1x+K_2}{\sqrt{2\pi}\,\sigma_X} \exp\left\{ \frac{-(x-\mu_X)^2}{2\sigma_X^2} \right\} \tag{C.6}$$

$$g(x) = (K_1x + K_2)p_X(x) \tag{C.7}$$

where constants K_1 and K_2 are:

$$K_1 = \frac{\sigma_\theta^2}{\sigma_\theta^2 + \sigma_N^2}, \qquad K_2 = \frac{\mu_\theta\sigma_N^2 - \mu_N\sigma_\theta^2}{\sigma_\theta^2 + \sigma_N^2}$$

Thus, equation (C.1) becomes:

$$y_i = K_1 \frac{\int_{x_i}^{x_{i+1}} x\, p_X(x)\, dx}{\int_{x_i}^{x_{i+1}} p_X(x)\, dx} + K_2, \qquad \forall i = 1,\cdots,N \tag{C.8}$$

Expanding equation (3.12):

$$\int_{-\infty}^{\infty} \theta^2\, p_N(x_i-\theta)p_X(x)dx - 2\, y_{i-1}\, g(x_i) + y_{i-1}^2\, p_X(x_i)$$
$$= \int_{-\infty}^{\infty} \theta^2\, p_N(x_i-\theta)p_X(x)dx - 2\, y_i\, g(x_i) + y_i^2\, p_X(x_i)$$

$$\Longrightarrow \quad 2\,(y_i - y_{i-1})\, g(x_i) = (y_i^2 - y_{i-1}^2)\, p_X(x_i) \tag{C.9}$$

Now,

$$g(x_i) = \int_{-\infty}^{\infty} \theta \, p_{\theta|X}(\theta|x_i) \, p_X(x_i) \, d\theta = K_1 x_i + K_2$$

$$K_1 x_i + K_2 = \frac{y_i + y_{i-1}}{2}, \qquad \forall i = 1, \cdots N \qquad \text{(C.10)}$$

And now equations (C.8) and (C.10) are the results quoted in equations (3.26) and (3.25), respectively.

C.2. Absolute Error Cost

The posterior density function for this example satisfies the conditions so that equation (3.17) may be used for when the cost function is the absolute error cost.

$$g(x_i) = \int_{-\infty}^{\infty} \theta \, p_{\theta|X}(\theta|x_i) \, p_X(x_i) \, d\theta = K_1 x_i + K_2$$

$$\Rightarrow g(x_i) = K_1 x_i + K_2 = \frac{y_i + y_{i-1}}{2}, \qquad \text{(C.11)}$$
$$\forall i = 1, \cdots N$$

And equation (3.19) may be used for the level condition:

$$\int_{x_i}^{x_{i+1}} \left[\int_{-\infty}^{y_i} p_{\theta|X}(\theta|x) \, d\theta \right] p_X(x) \, dx = \frac{1}{2} \int_{xi}^{x_{i+1}} p_X(x) \, dx$$

which from (C.5) implies:

$$\int_{x_i}^{x_{i+1}} \text{erf} \left[\frac{y_i - g(x)}{(\sigma_\theta^2 \sigma_N^2 / (\sigma_\theta^2 + \sigma_N^2))^{.5}} \right] p_X(x) \, dx = \frac{1}{2} \int_{xi}^{x_{i+1}} p_X(x) \, dx \qquad \text{(C.12)}$$

C.3. Uniform Error Cost

The posterior density function satisfies the necessary conditions so that equation (3.22) may be used for the threshold condition and the uniform error cost function.

$$g(x_i) = \int_{-\infty}^{\infty} \theta \, p_{\theta|X}(\theta|x_i) \, p_X(x_i) \, d\theta = K_1 x_i + K_2$$

$$\Rightarrow g(x_i) = K_1 x_i + K_2 = \frac{y_i + y_{i-1}}{2}, \quad \forall i = 1, \cdots N \qquad \text{(C.13)}$$

Likewise, the level condition of equation (3.24) may be used:

$$p_\theta(y_i - \Delta/2) \int_{x_i}^{x_{i+1}} p_N(x-(y_i-\Delta/2)) \, dx = p_\theta(y_i+\Delta/2) \int_{x_i}^{x_{i+1}} p_N(x-(y_i+\Delta/2)) \, dx \qquad \text{(C.14)}$$

By evaluating (C.14) with the Gaussian probability density functions for the example:

$$\exp(\Delta(y_i - \mu_\theta)/2\sigma_\theta^2) \left\{ \mathrm{erf}\left[\frac{x_{i+1}-(y_i+\Delta/2)-\mu_N}{\sigma_N}\right] - \mathrm{erf}\left[\frac{x_i-(y_i+\Delta/2)-\mu_N}{\sigma_N}\right] \right\}$$
$$= \exp(-\Delta(y_i - \mu_\theta)/2\sigma_\theta^2) \left\{ \mathrm{erf}\left[\frac{x_{i+1}-(y_i-\Delta/2)-\mu_N}{\sigma_N}\right] - \mathrm{erf}\left[\frac{x_i-(y_i-\Delta/2)-\mu_N}{\sigma_N}\right] \right\} \qquad \text{(C.15)}$$

APPENDIX D

Pictures

APPENDIX D

Images

The following pages contain sample pictures of the quantization of a noisy image source with the Max-Lloyd, uniform, and minimum risk quantizers as described in chapter 5. The source is assumed Gaussian with the *a priori* statistics obtained by sampling a window placed about the face of the girl in the image. These statistics were: μ_θ = 101.71, and σ_θ = 25.1 on a 256-level intensity gray scale. The quantizers were obtained by scaling by σ_θ and shifting by μ_θ a Max-Lloyd (with squared-error distortion function) quantizer designed for a source ~ N(0,1). The is based on the conversion of any random variable from a normal density function to a standard normal density through:

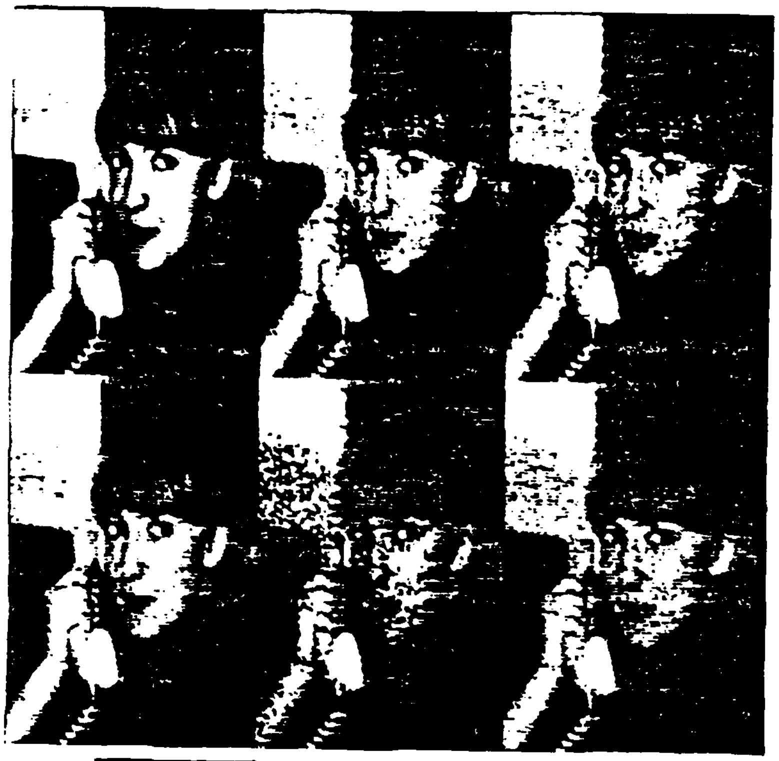
$$z = \frac{(x - \mu)}{s},$$

$$x \sim N(\mu, s),$$

$$z \sim N(0,1)$$

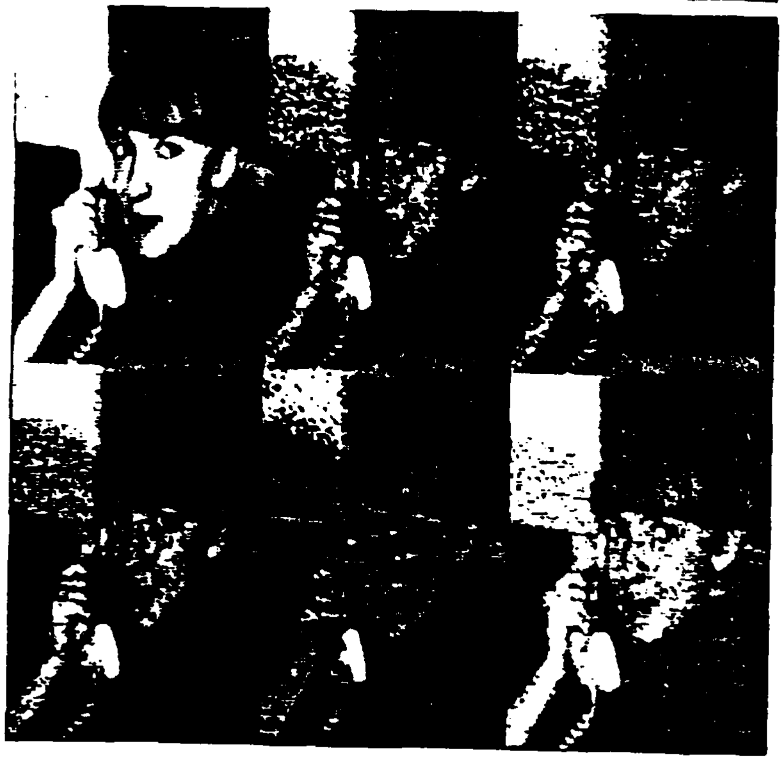
Note how at the lower SNR rates, the minimum-risk quantizer levels converge to the mean of the source.

This is the reason for the "washed-out" or low contrast appearance. Also note how the noise mean adversely affects all of the quantizers with the exception of the minimum risk. While this may not be the ideal demonstration of the quantization, the results do indicate the degree of success of quantizing noisy sources relative to other approaches.



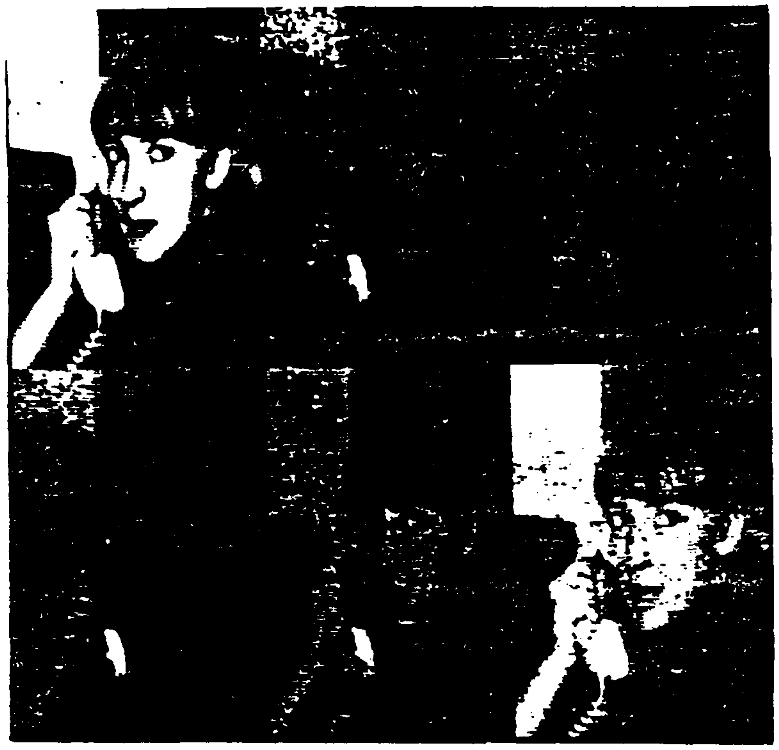
$\mu_V = 0$

S/N = 8 dB



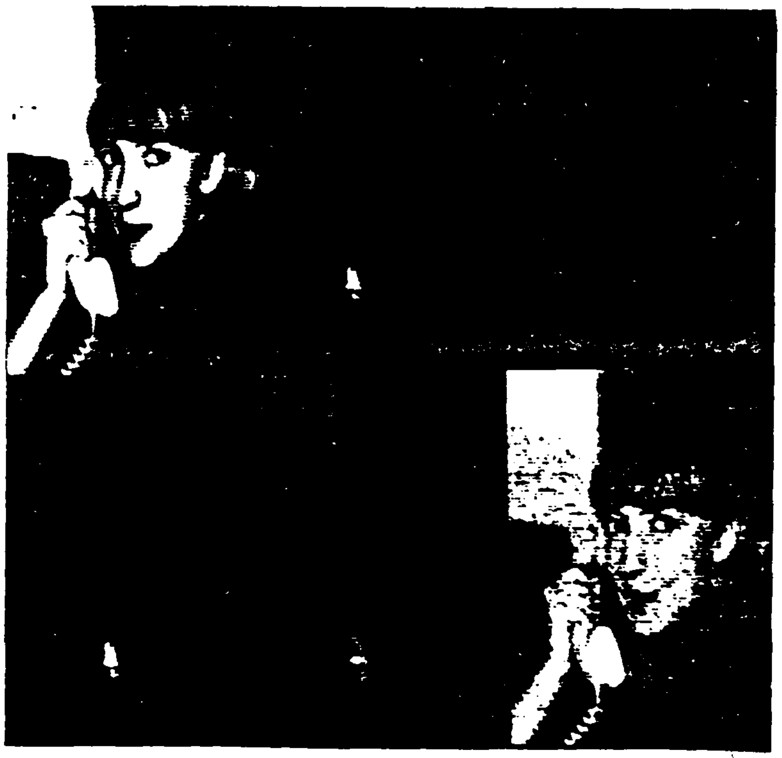
$\mu_N = -25$

S/N = 8 dB



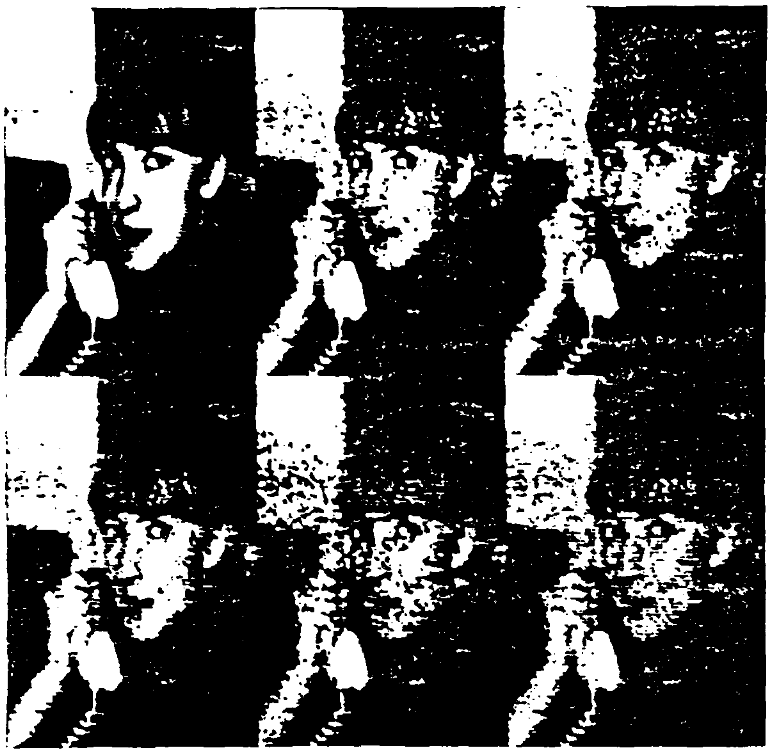
$\mu_N = -50$

$S/N = 8$ dB



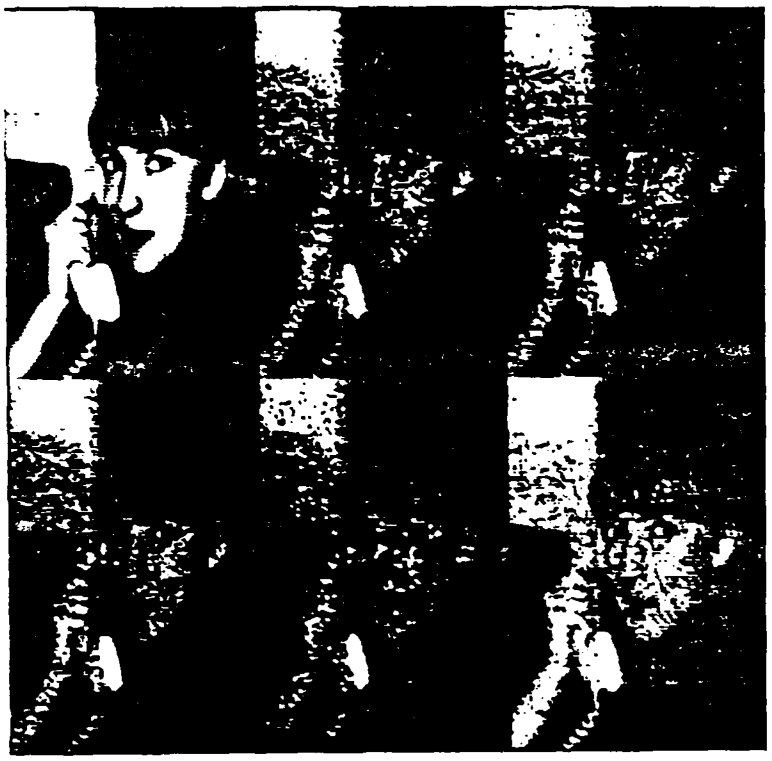
$\mu_N = -75$

$S/N = 8$ dB



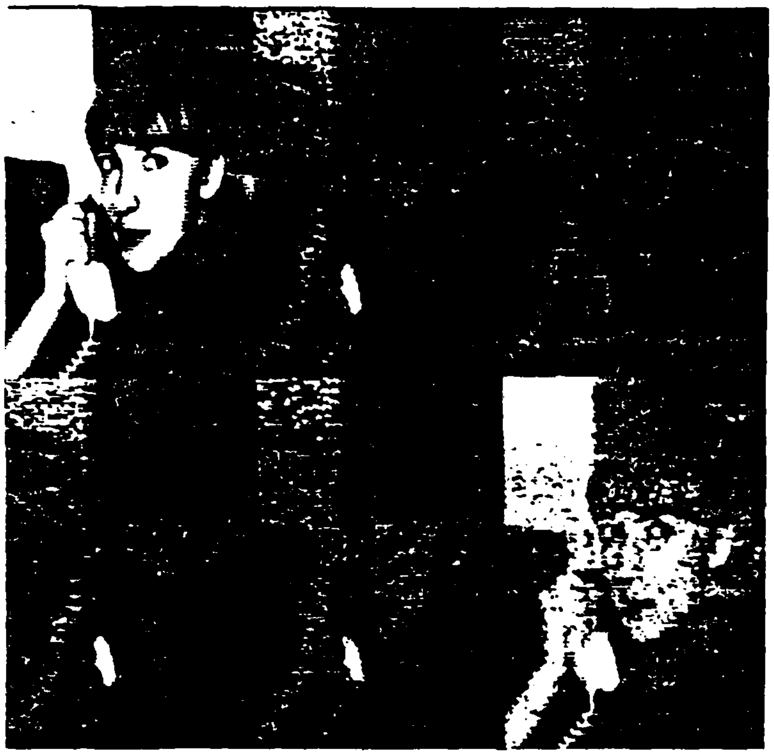
$\mu_N = 0$

$S/N = 4$ dB



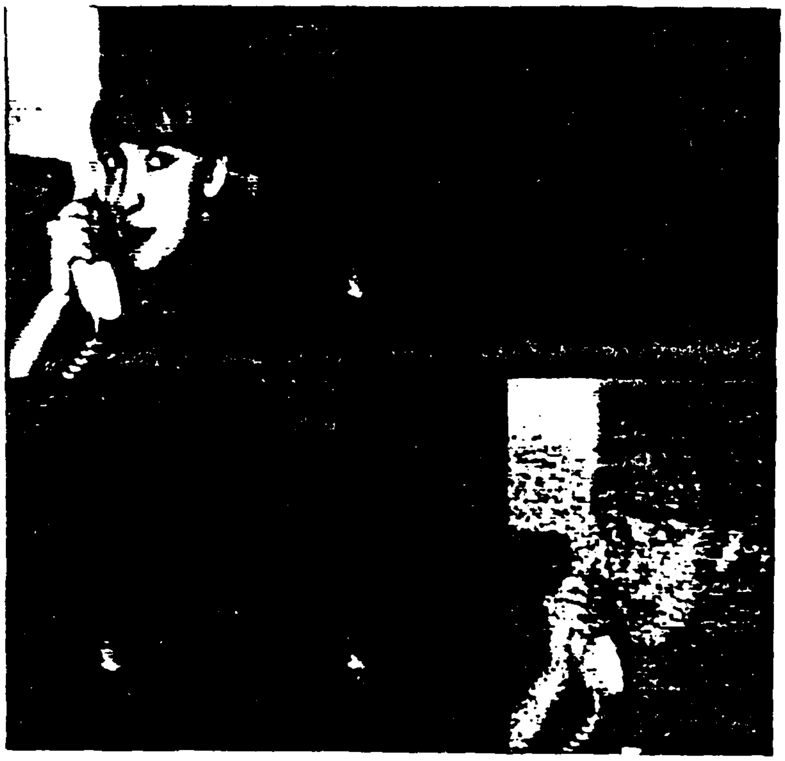
$\mu_N = -25$

$S/N = 4$ dB



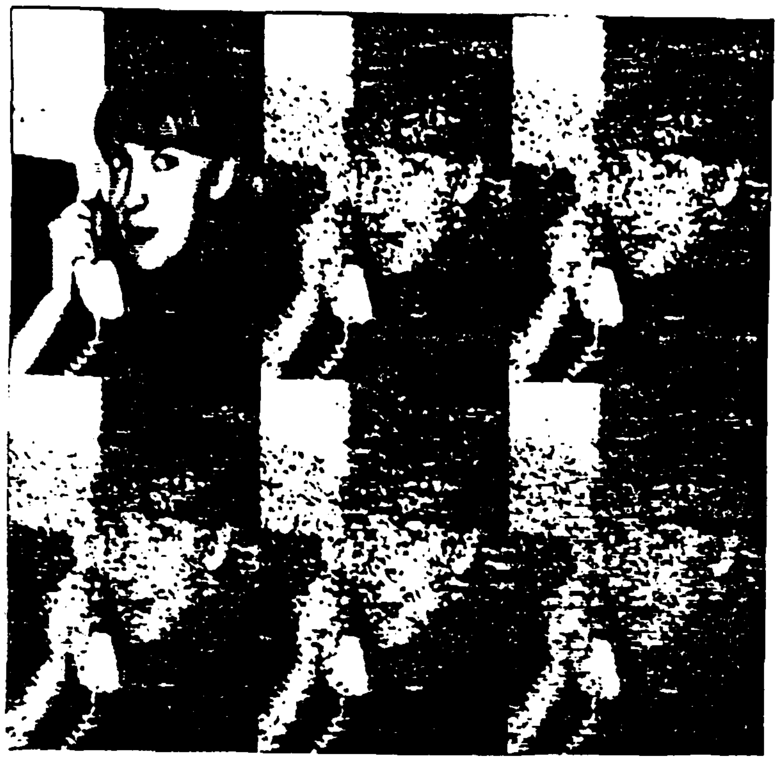
$\mu_V = -50$

$S/N = 4$ dB



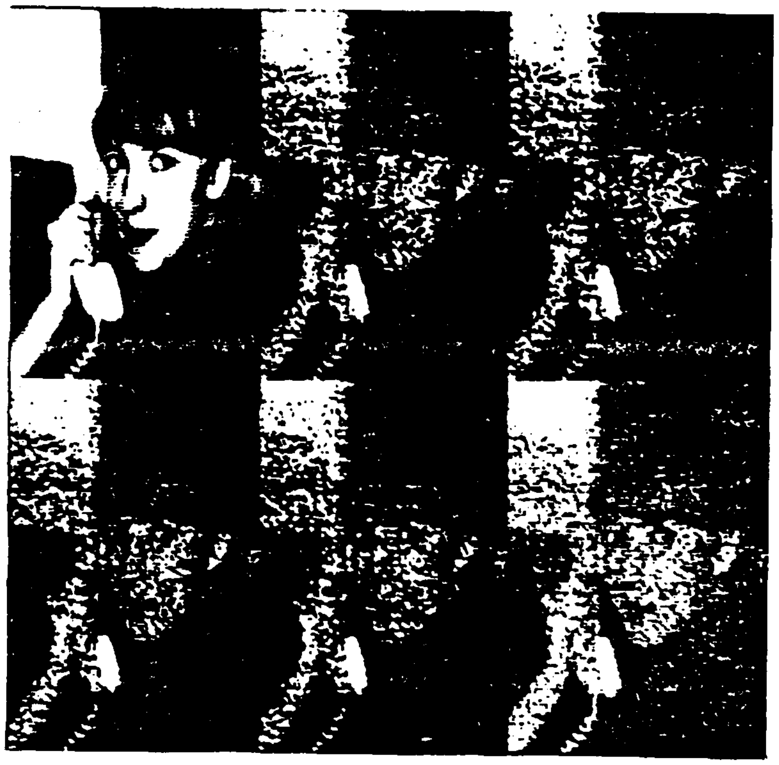
$\mu_V = -75$

$S/N = 4$ dB



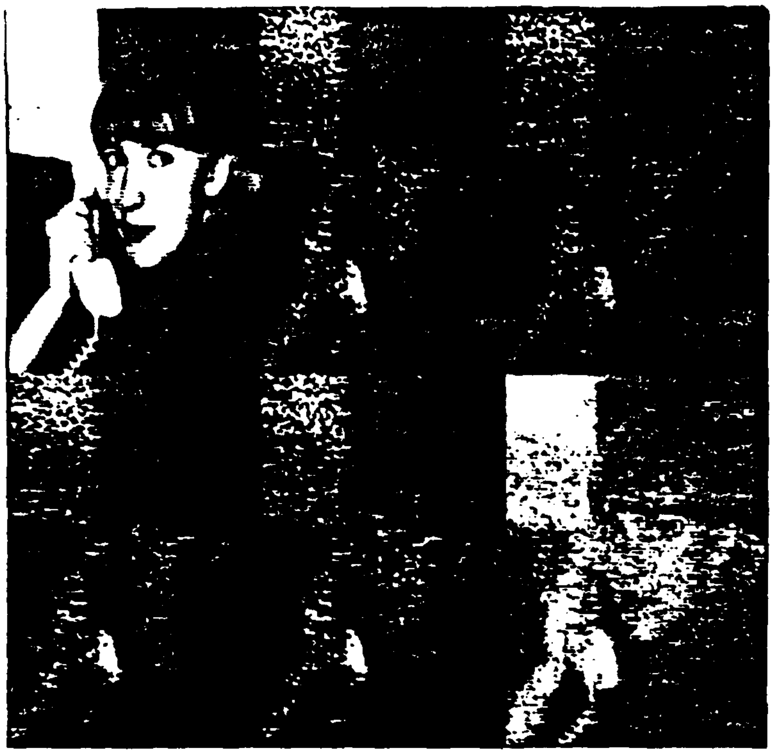
$\mu_V = 0$

S/N = 0 dB



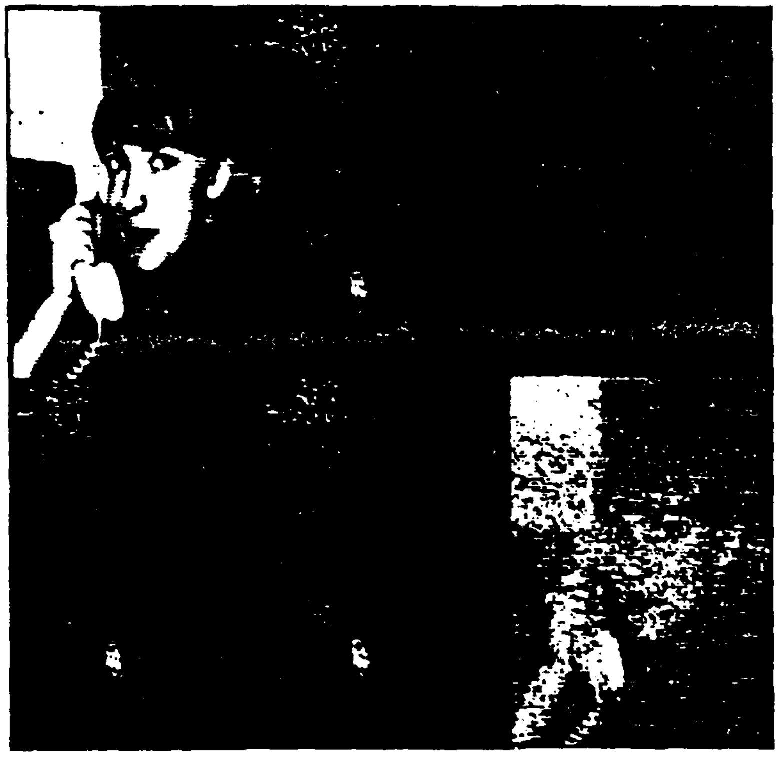
$\mu_V = -25$

S/N = 0 dB



$\mu_N = -50$

S/N = 0 dB



$\mu_N = -75$

S/N = 0 dB

APPENDIX E

Computer Programs

```
/*--------------- Quantizer.h -----------------*/

#define QMAX 128
#define LEVEL(x,n) *(x->level+n)
#define THRESH(x,n) *(x->thresh+n)
#define NLEVELS(x) x->num_levels

typedef struct quantizer {
                          int num_levels;
                          double *level;
                          double *thresh;
                 } QUANT, *QUANTPTR;

/*--------------- Function.h ------------------*/

#define INTERVALS 5000
#define INFINITY 10.
#define FMAX 10
#define FPARM(x) x->parameter
#define FDPARM(x,n) (x->parameter+n)->dval
#define FFPARM(x,n) (x->parameter+n)->function
#define FFUNCT(x,n) (x->parameter+n)->fnct
#define FNUM(x) x->num_parameters

struct funct {
               union parm {
                            double dval;
                            double (*function) ();
                            struct funct *fnct;
                          }  *parameter;
               int num_parameters;
             };

typedef union parm PARM, *PARMPTR;
typedef struct funct FUNCT, *FUNCTPTR;
```

```
/*---------------- Function.c -----------------*/

#include <stdio.h>
#include <stdlib.h>
#include <function.h>

FUNCTPTR f_alloc(n)
int n;
{
FUNCTPTR f;
  f = (FUNCTPTR) calloc(1,sizeof(FUNCT));
  FNUM(f) = n;
  FPARM(f) = (PARMPTR) calloc(FMAX,sizeof(PARM));
  return(f);
}


int free_funct(f)
FUNCTPTR f;
{
extern int free();
  free(FPARM(f));
  free(f);
}
```

```
/*------------ Quantizer.c (Newquant) ----*/

#include <stdio.h>
#include <stdlib.h>
#include <quantizer.h>

QUANTPTR q_alloc(n)
int n;
{
QUANTPTR quant;
  quant = (QUANTPTR) calloc(1,sizeof(QUANT));
  NLEVELS(quant) = n;
  quant->level = (double *) calloc(QMAX,sizeof(double));
  quant->thresh = (double *) calloc(QMAX,sizeof(double));
  return(quant);
}


int zero_quant(q)
QUANTPTR q;
{
int i;

  for(i = 0; i <= NLEVELS(q); ++i) {
    THRESH(q,i) = 0.;
    LEVEL(q,i) = 0.;
  }
}


int print_quantizer(q)
QUANTPTR q;
{
int i,n;
  n = NLEVELS(q);
  for(i=0; i<n; ++i)
   printf("%lf  %lf\n",THRESH(q,i),LEVEL(q,i));
}
```

```
int fprint_quantizer(q,fp)
QUANTPTR q;
FILE *fp;
{
int i,n;
  n = NLEVELS(q);
  for(i=0; i<n; ++i)
   fprintf(fp,"%lf  %lf\n",THRESH(q,i),LEVEL(q,i));
}


int free_quant(q)
QUANTPTR q;
{
extern int free();

  free(q->thresh);
  free(q->level);
  free(q);
}
```

```
/*---------------------------------------------------
  Function to quantize a value with a given quantizer.

  Quantizer is assumed to have the form:


   --         --
   : x1     y1 :       where level yj corresponds to
   : x2     y2 :       an x in the region (xj,xj+1).
   :  .      . :
   : xn     yn :       also xn+1 is assumed +infinity,
   --         --       and no value is stored for it.


---------------------------------------------------*/


double quantize(value,q)
double value;
QUANTPTR q;
{
int ptr = 0;
  while ( (value > THRESH(q,ptr)) && (ptr < NLEVELS(q)) )
    ptr++;
  ptr--;
  return( LEVEL(q,ptr) );
}

/*--------------- END of Quantizer.c ---------------*/

/*------------------ Integral.c ------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <function.h>


double dabs(x)
double x;
{
  if ( x < 0.)
    return( -x );
  else
```

```
    return( x );
}


double integral(xl,xu,f)
double xl;
double xu;
FUNCTPTR f;
{
extern double intdiv(),dabs();
double areal,area,diff,tmp,eps = 1.;
int n;

  if (xl > xu) {
    tmp = xl;           /* Want to integral from lower value */
    xl = xu;            /* to the higher value.              */
    xu = tmp;
  }
  areal = intdiv(xl,xu,f,1);
  n = 2;
  area = intdiv(xl,xu,f,n);
  while( (dabs(area) > 0.0001)
/*      && (dabs(diff = area - areal) > 0.0001)
*/
        && ((dabs((diff = area - areal) / area) * 100.) > eps) ) {
    areal = area;
    n++;
    area = intdiv(xl,xu,f,n);
/* printf("n:%d area:%lf\n",n,area);
*/
  }
  return( area );
}


double intdiv(xl,xu,f,n)
double xl;
double xu;
FUNCTPTR f;
int n;
{
double intgrl();
double lower,upper,incr,area = 0.;
```

```
int i;
  incr = (xu - xl) / n;
  lower = xl;
  for(i = 0; i < n; ++i) {
    upper = lower + incr;
    area += intgrl(lower,upper,f);
    lower = upper;
  }
  return( area );
}



double intgrl(xl,xu,f)
double xl;
double xu;
FUNCTPTR f;
{
double (*fct) ();
double a,b,xp,xm,c,y;
      fct = FFPARM(f,0);
      a = 0.5E+0 * (xu+xl);
      b = xu-xl;
      c = 0.49863193092474078E+0 * b;
      FDPARM(f,1) = a + c;
      xp = (*fct) (f);
      FDPARM(f,1) = a - c;
      xm = (*fct) (f);
      y = 0.35093050047350483E-2 * (xp + xm);
      c = 0.49280575577263417E+0 * b;
      FDPARM(f,1) = a + c;
      xp = (*fct) (f);
      FDPARM(f,1) = a - c;
      xm = (*fct) (f);
      y = y+0.81371973654528350E-2 * (xp + xm);
      c = 0.48238112779375322E+0 * b;
      FDPARM(f,1) = a + c;
      xp = (*fct) (f);
      FDPARM(f,1) = a - c;
      xm = (*fct) (f);
      y = y+0.12696032654631030E-1 * (xp + xm);
      c = 0.46745303796886984E+0 * b;
      FDPARM(f,1) = a + c;
```

```
xp = (*fct) (f);
FDPARM(f,1) = a - c;
xm = (*fct) (f);
y = y+0.17136931456510717E-1 * (xp + xm);
c = 0.44816057788302606E+0 * b;
FDPARM(f,1) = a + c;
xp = (*fct) (f);
FDPARM(f,1) = a - c;
xm = (*fct) (f);
y = y+0.21417949011113340E-1 * (xp + xm);
c = 0.42468380686628499E+0 * b;
FDPARM(f,1) = a + c;
xp = (*fct) (f);
FDPARM(f,1) = a - c;
xm = (*fct) (f);
y = y+0.25499029631188088E-1 * (xp + xm);
c = 0.39724189798397120E+0 * b;
FDPARM(f,1) = a + c;
xp = (*fct) (f);
FDPARM(f,1) = a - c;
xm = (*fct) (f);
y = y+0.29342046739267774E-1 * (xp + xm);
c = 0.36609105937014484E+0 * b;
FDPARM(f,1) = a + c;
xp = (*fct) (f);
FDPARM(f,1) = a - c;
xm = (*fct) (f);
y = y+0.32911111388180923E-1 * (xp + xm);
c = 0.33152213346510760E+0 * b;
FDPARM(f,1) = a + c;
xp = (*fct) (f);
FDPARM(f,1) = a - c;
xm = (*fct) (f);
y = y+0.36172897054424253E-1 * (xp + xm);
c = 0.29385787862038116E+0 * b;
FDPARM(f,1) = a + c;
xp = (*fct) (f);
FDPARM(f,1) = a - c;
xm = (*fct) (f);
y = y+0.39096947893535153E-1 * (xp + xm);
c = 0.25344995446611470E+0 * b;
FDPARM(f,1) = a + c;
xp = (*fct) (f);
```

```
        FDPARM(f,1) = a - c;
        xm = (*fct) (f);
        y = y+0.41655962113473378E-1 * (xp + xm);
        c = 0.21067563806531767E+0 * b;
        FDPARM(f,1) = a + c;
        xp = (*fct) (f);
        FDPARM(f,1) = a - c;
        xm = (*fct) (f);
        y = y+0.43826046502201906E-1 * (xp + xm);
        c = 0.16593430114106382E+0 * b;
        FDPARM(f,1) = a + c;
        xp = (*fct) (f);
        FDPARM(f,1) = a - c;
        xm = (*fct) (f);
        y = y+0.45586939347881942E-1 * (xp + xm);
        c = 0.11964368112606854E+0 * b;
        FDPARM(f,1) = a + c;
        xp = (*fct) (f);
        FDPARM(f,1) = a - c;
        xm = (*fct) (f);
        y = y+0.46922199540402283E-1 * (xp + xm);
        c = 0.7223598079139825E-1 * b;
        FDPARM(f,1) = a + c;
        xp = (*fct) (f);
        FDPARM(f,1) = a - c;
        xm = (*fct) (f);
        y = y+0.47819360039637430E-1 * (xp + xm);
        c = 0.24153832843869158E-1 * b;
        FDPARM(f,1) = a + c;
        xp = (*fct) (f);
        FDPARM(f,1) = a - c;
        xm = (*fct) (f);
        y = b * (y+0.48270044257363900E-1 * (xp + xm));
        return (y);
}
```

```
/*-----------------------------------------------------------

      Minimax-Risk Program

This program is intended to determine the minimax risk
quantizer for a known Gaussian p.d.f.  It uses determines
mines the Lagrange multipliers needed for the maximization
of the I(noise) function, such that that maximum is zero
and the Lagrange multipliers are greather than or equal
to zero.

-----------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>


#define INFINITY 30000.0


/*-----------------------------------------------------------

      Variable description:

         C - Cost function
         qthr,mthr - quantizer threshold set
         qlev,mlev - quantizer level set
         tl,tu - threshold lower and upper limits
         nl,nstep,nu - noise limits and increment
         d2min - fidelity criterion for lambda2 = .00001
         d2max - maximum expected lambda2
         l20 - minimum expected lambda2 = 0
         constraint - on mean square power of the noise
         mmrisk - minimax risk
         limit - bound on the distortion measure
         lambda1, lambda2 - Lagrange multipliers
         lower,step,upper - limits and step on theta
         mean, std - on theta

-----------------------------------------------------------*/
```

```
#define C(theta,y) (theta - y) * (theta - y)
#define EVER (;;)
#define N 4

#define max(a,b)        (((a) > (b)) ? (a) : (b))
#define min(a,b)        (((a) < (b)) ? (a) : (b))
#define abs(a)          (((a) < 0.0) ? -(a) : (a))

FILE *out;

float qthr[5], qlev[4];
float mthr[5], mlev[4];
float tl,tu,md1,md2,md3;
float nl,nstep,nu;             /* limits on noise */
float mean, std, twostd;
float d2min, d2max, l20;
float constraint, limit;
float lambda1, lambda2;
float mmrisk;                  /* Optimum minimax risk */
float lower,step,upper;


/*---------- Gaussian p.d.f. of N(mean,std) -------------*/


float p_normal(x)
float x;
{
      extern float mean, std;
      extern double exp();
      float arg;

      arg = -(x-mean)*(x-mean)/(2.*std*std);

      if (arg < -30.0)
         return(0.0);
      else
         return( 0.3989422804 * exp((double) arg) / std);
}
```

```
int print_quantizer()
{
      extern float qthr[], qlev[];
      int i;

     for(i = 0; i < N; i++)
         printf("thr: %e   lev: %e\n",qthr[i],qlev[i]);
      printf("thr: %e\n",qthr[N]);
}


int fprint_quantizer()
{
      extern FILE *out;
      extern float d2min,d2max,l20,constraint,mmrisk;
      extern float std,limit;
      extern float qthr[], qlev[];
      extern float tl,step,tu,nl,nstep,nu;
      int i;

      fprintf(out,"std: %e\n",std);
      fprintf(out,"d2min: %e  d2max: %e\n",d2min,d2max);
      fprintf(out,"l20: %e\n",l20);
      fprintf(out,"tu: %e nu: %e\n",tu,nu);
      fprintf(out,"constraint: %e\n",constraint);
      fprintf(out,"limit: %e\n",limit);

     for(i = 0; i < N; i++)
        fprintf(out,"thr: %e lev: %e\n",qthr[i],qlev[i]);

      fprintf(out,"thr: %e\n",qthr[N]);

      fprintf(out,"mmrisk: %e\n",mmrisk);
}
```

```
/*-- Writes out an intermediate quant. as it finds them --*/

int ffprint_quantizer()
{
      extern FILE *fopen();
      extern float mmrisk,d2min,d2max,120;
      extern float constraint,std,limit;
      extern float qthr[], qlev[];
      extern float tu, nu;
      FILE *output;
      char outname[50];
      int i;

      strcpy(outname,"mm40out XXXXXXXX b");
      mktemp(outname);
      output = fopen(outname,"w");

      fprintf(output,"tu: %e nu: %e\n",tu,nu);
      fprintf(output,"d2min:%e d2max:%e\n",d2min,d2max);
      fprintf(output,"120: %e\n",120);
      fprintf(output,"std: %e\n",std);
      fprintf(output,"constraint: %e\n",constraint);
      fprintf(output,"limit: %e\n",limit);

     for(i = 0; i < N; i++)
         fprintf(output,"thr: %e lev: %e\n",qthr[i],qlev[i]);
      fprintf(output,"thr: %e\n",qthr[N]);

      fprintf(output,"mmrisk: %e\n",mmrisk);
      fclose(output);
}
```

```
/*---- Stores the current quantizer as minimax ----------*/

int copy_quantizer()
{
      extern float qthr[],qlev[],mthr[],mlev[];
      int i;

      for(i = 0; i < N; i++) {
         mthr[i] = qthr[i];
         mlev[i] = qlev[i];
      }
}



/*----- Quantizes a sample ---------------------------*/

float q(x)
float x;
{
      extern float qthr[], qlev[];
      register int k = 0;

      while( (k < (N - 1)) && (qthr[k+1] <= x) )
         k++;
      if(qthr[N] <= x)
         return(qlev[N-1]);
      else
           return(qlev[k]);
}
```

```
/*----- Finds maximum of l1(l2) over the noise range -----*/


float xxeval(noise)
float noise;
{
      extern float p_normal(), lambda2, mean, limit;
      extern float qthr[], qlev[], lower, step, upper;
      float i, theta;
      float x, q();
      float sum,val;

         sum = 0.0;
         for(theta = lower; theta <= upper; theta += step )  {
            x = theta + noise;
            if((val = C(theta,q(x))) > limit)
               sum += limit * p_normal(theta);
            else
               sum += val * p_normal(theta);
/*
printf("n:%e t:%e val:%e\n",noise,theta,val);
*/
         }
/*
printf("sum:%e\n",sum);
*/
      sum *= step;                 /* Riemann sum * step */
      sum -= lambda2 * noise * noise;
/*
printf("noise: %e  final sum:%e\n",noise,sum);
*/
      return(sum);
}
```

```
/*------------ Determines lambdal(lambda2) -------------*/

float l1(l2)
float l2;
{
   extern float lambdal, lambda2, mean;
   extern float nl,nstep,nu;
   float noise;
   float xxeval(), val;

   lambdal = -INFINITY;
   lambda2 = l2;

   for(noise = nl; noise <= nu; noise += nstep) {
      val = xxeval(noise);
      lambdal = (val > lambdal) ? val : lambdal;
/*
printf("noise:%e  val:%e\n",noise,val);
*/
   }
   return(lambdal);
}


/*------ Performs 5-point minimization over lambda2 ------*/
/*------ as outlined in Bath's dissertation.        ------*/


float l2()
{
      extern float constraint, lambdal, l20, d2max, d2min;
      float j[4], l2, d2, l2star, l1();
      int k;

      l2star = l20;
      d2 = d2max;

      while(d2 > d2min) {
         for(k = 0; k < 4; k++) {
            l2 = l2star + k * d2;
            j[k] = l1(l2) + l2 * constraint;
/*
printf("l2star: %e  l2: %e  j[%d]: %e  d2: %e\n",
l2star,l2,k,j[k],d2);
*/
            }
            if (j[0] <= j[1])
               d2 /= 4.0;
            if((j[0] > j[1]) && (j[1] <= j[2]))
               d2 /= 2.0;
            if((j[0] > j[1]) && (j[1] > j[2]) && (j[2] <= j[3])) {
               l2star += d2;
               d2 /= 2.0;
            }
            if((j[0] > j[1]) && (j[1] > j[2]) && (j[2] > j[3])) {
               l2star += 2.0 * d2;
               d2 /= 2.0;
            }
         }

         return(l1(l2star) + l2star * constraint);
}
```

```
/*---- Determines the minimax risk quantizer -----------*/

float riskq()
{
        extern FILE *out;
        extern float tl,nstep, tu;
        extern float md1, md2, md3;
        extern float qthr[], qlev[];
        extern float constraint, mean;
        extern float mmrisk;
        extern float lambda1, lambda2;
        int copy_quantizer(), fprint_quantizer(), print_quantizer();
        int ffprint_quantizer();
        float end, tlp1, tlp2;
        int flthr, fllev;
        float l2();
        float curr_risk, last;

        mmrisk = INFINITY;

    tlp1 = tl + nstep;
    tlp2 = tl + 2. * nstep;

    for(qthr[1] = md2; qthr[1] > tlp2; qthr[1] -= nstep) {
     for(qlev[0] = md3; qlev[0] > tlp1; qlev[0] -= nstep) {
      fllev = 0;
      last = INFINITY;    /* restart risk calc for lev0-- */
      end = qlev[0];
      for(qlev[1] = md1; qlev[1] > end; qlev[1] -= nstep) {

                qthr[3] = qthr[2] + (qthr[2] - qthr[1]);
                qlev[2] = qthr[2] + (qthr[2] - qlev[1]);
                qlev[3] = qthr[2] + (qthr[2] - qlev[0]);
/*
                print_quantizer();
*/
                curr_risk = l2();

                if(curr_risk > last) {
                   qlev[1] = end;
                }
```

```
                    last = curr_risk;

                    if(curr_risk <= mmrisk) {
                       flthr = 1;              /* mmrisk for this thr */
                       copy_quantizer();
                       md3 = qlev[0];        /* new start for outer lev*/
                       fprint_quantizer();
                       mmrisk = curr_risk;
                     printf("mmrisk: %le\n", mmrisk);
                     print_quantizer();
                       fprintf(out,"mmrisk: %e\n",mmrisk);

                       ffprint_quantizer();

                    }
            }
        }
        }
}


/*------ Evaluates the risk for a specific quantizer -----*/


float calc_risk()
{
      extern float constraint;
      extern float qthr[], qlev[];
      extern float mmrisk;
      extern float mean;
      extern int fprint_quantizer(), print_quantizer();
      float q(),x;
      float l2();
      int i;

for EVER {
mmrisk = INFINITY;
   for(i = 0; i < 2; i++) {
     printf("Enter thr[%d], level[%d]: ",i,i);
     scanf("%e %e",&qthr[i],&qlev[i]);
   }
```

```
    qthr[2] = mean;
    qthr[3] = qthr[2] + (qthr[2] - qthr[1]);
    qthr[4] = qthr[2] + (qthr[2] - qthr[0]);
    qlev[2] = qthr[2] + (qthr[2] - qlev[1]);
    qlev[3] = qthr[2] + (qthr[2] - qlev[0]);



        mmrisk = 12();

        print_quantizer();
        printf("mmrisk: %e\n",mmrisk);
    }

    return(mmrisk);
}


/*---------- Main driver program ------------------------*/


main()
{
    extern FILE *out;
    extern float tl, tu, mthr[], mlev[];
    extern float nl, nstep, nu;
    extern float constraint, mean, std, twostd;
    extern float limit;
    extern float d2min, d2max, 120;
    extern float lower, step, upper, md1, md2, md3;
    extern double sqrt();
    char oname[50];
    int fprint_quantizer(), print_quantizer(),i;
    int ffprint_quantizer();
    float calc_risk(), riskq();

    printf("Minimax-Risk quantizer program.\n\n");

    printf("Enter output filename:");
    gets(oname);

    out = fopen(oname,"w");
```

```
        printf("Enter tu (mean will be added): ");
        scanf("%e",&tu);

        printf("Enter source mean and std:");
        scanf("%e %e",&mean,&std);

        tl = mean - tu;
        tu = mean + tu;

        twostd = 2.0 * std;

        lower = mean - twostd;
        upper = mean + twostd;
        step = (upper - lower) / 25.;

printf("tl:%e tu:%e 2std:%e lower:%e step:%e upper:%e\n",
tl,step,tu,twostd,lower,step,upper);

        printf("Enter constraint: ");
        scanf("%e",&constraint);
        printf("constraint:%e\n",constraint);

        nu = mean + 2.0 * sqrt((double) constraint);
        nl = mean;
        nstep = (nu - nl) / 25.;

        printf("nl:%e nstep:%e nu:\n",nl,nstep,nu);

        printf("Enter bound on cost: ");
        scanf("%e",&limit);
        printf("limit:%e\n",limit);

        printf("Enter d2min, d2max: ");
        scanf("%e %e",&d2min,&d2max);

        printf("Enter 120: ");
        scanf("%e",&120);

        printf("Starting quantizer\n");
        printf("Enter Levell Thresh Level2 (w.r.t. mean):");
        scanf("%e %e %e",&md1,&md2,&md3);
```

```
/* Assume a symmetric quantizer */
      qthr[0] = tl;
      qthr[2] = mean;
      qthr[4] = tu;

      md1 = qthr[2] - md1;
      md2 = qthr[2] - md2;
      md3 = qthr[2] - md3;

      printf("d2min: %e  d2max: %e\n",d2min,d2max);
      printf("120: %e \n",120);
      printf("limit: %e\n",limit);

      fprintf(out,"limit: %e\n",limit);
      fprintf(out,"d2min: %e  d2max: %e\n",d2min,d2max);
      fprintf(out,"120: %e\n",120);

/* Replace with calc_risk for a specific quantizer testing */

      riskq();

      printf("Minimax-risk: %e c:%e\n",mmrisk,constraint);

      for(i = 0; i < N; i++)
         printf("thr: %e   lev: %e\n",mthr[i],mlev[i]);

      fprintf(out,"Minimax-risk: %e c:%e\n",mmrisk,constraint);

      for(i = 0; i < N; i++)
         fprintf(out,"thr: %e   lev: %e\n",mthr[i],mlev[i]);

      fclose(out);
      exit(0);
}
```

```
/*-----------------------------------------------------------
     RABS11.C

Program to determine the minimum risk quantizer for the
absolute-error cost functions and Gaussian source + Gaussian
noise.

-----------------------------------------------------------*/


#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <function.h>
#include <quantizer.h>


/*-----------------------------------------------------------

      Variable descriptions:

         q - quantizer
         n_mean, n_std - noise statistics
         x_mean, x_std - source statistics
         r_mean, r_std - observation statistics
         n_inf,x_inf,r_inf - "infinities for resp. densities

-----------------------------------------------------------*/

#define INCREMENT .1
#define FIDELITY .00001

QUANTPTR q;
double increment = INCREMENT;
double n_mean,n_std,x_mean,x_std,r_mean,r_std;
double x_inf,n_inf,r_inf;
double k1,k2,x_var,n_var,r_var;
double sgn(x)
double x;
{
  if(x < 0.)
    return (-1.);
  else if(x == 0.)
    return (0.);
  else
    return (1.);
}


/*--- Gaussian p.d.f. for N(mean,std) -------------------*/


double p_normal(mean,std,x)
double mean;
double std;
double x;
{
extern double exp();
double arg;
  if ((x > (5.0 * std + mean)) || (x < (-5.0 * std + mean)))
    return(0.);
  else {
    arg = -(x-mean)*(x-mean)/(2.*std*std);
    return( 0.3989422804 * exp(arg) / std);
  }
}


/*--- Integrable p.d.f function --------------------------*/

double pnorm(f)
FUNCTPTR f;
{
extern double p_normal();
double arg;

  arg = FDPARM(f,1);
  return( p_normal(0.,1.,arg) );

}
```

```
double p_noise(n)
double n;
{
extern double n_mean, n_std, p_normal();
    return(p_normal(n_mean,n_std,n));
}


double p_x(x)
double x;
{
extern double x_mean, x_std, p_normal();
  return(p_normal(x_mean,x_std,x));
}


/*---------------------------------------------------------------
  Observation probability density function.                    :
   Defined as the convolution of p(x) and p(n) at              :
   the observation point r.                                    */


double p_r(f)
FUNCTPTR f;
{
extern double p_normal(), r_mean, r_std;
double r;

  r = FDPARM(f,1);
  return(p_normal(r_mean,r_std,r));
}


/*------- Initialize quantizer -------------------------*/


int sset_initial(q,guess)
QUANTPTR q;
double guess;
{
extern double r_mean, r_inf;

  THRESH(q,0) = 101.71;
```

```
    LEVEL(q,0) = 102.0301631;
    THRESH(q,1) = 103.0981765;
    LEVEL(q,1) = 102.7780127;

}


int set_initial(q,guess)
QUANTPTR q;
double guess;
{
extern double r_mean, r_inf;

  THRESH(q,0) = r_mean;
  LEVEL(q,0) = guess;
  THRESH(q,1) = r_mean + r_std;

}



double db(x)
double x;
{
extern double log10();
  return(10. * log10(x));
}


double dbinv(x)
double x;
{
extern double pow();
  return( pow(10., 0.1 * x) );
}
```

```
/*----- Detetermine if quantizer criteria are met --------*/


double abscl(rj,yj,rjpl)
double rj;
double yj;
double rjpl;
{
extern FUNCTPTR f_alloc();
extern double abscla(),p_r(),cum_norm(),intgrl();
extern double kl,k2,r_mean,r_std;
FUNCTPTR f;
double a,b,xl,xu;
  f = f_alloc(10);
  FFPARM(f,0) = abscla;
  FDPARM(f,2) = yj;
  a = intgrl(rj,rjpl,f);
  free_funct(f);
  xl = (rj - r_mean) / r_std;          /* find prob of p(r) between */
  xu = (rjpl - r_mean) / r_std;        /* rj and rj+l  (gaussian so */
  b = (cum_norm(xu)-cum_norm(xl))/2.;  /* can use F(x) instead of   */
/*
  printf("a:%lf b:%lf rj:%lf yj:%lf rjpl:%lf\n",a,b,rj,yj,rjpl);
*/
  return(a - b);                       /* general integration.      */
}


double abscla(f)
FUNCTPTR f;
{
extern FUNCTPTR f_alloc();
extern double p_r(),cum_norm(),sqrt();
extern double kl,k2,n_var;
extern int free_funct();
FUNCTPTR fp;
double r, yj, pr_r, mean, std;

  fp = f_alloc(10);
  r = FDPARM(f,1);
  FDPARM(fp,1) = r;
  yj = FDPARM(f,2);
  pr_r = p_r(fp);
```

```
   free_funct(fp);
   mean = k1 * r + k2;
   std = sqrt( n_var * k1);
   return( cum_norm( (yj-mean)/std ) * pr_r );
}


double cum_norm(x)
double x;
{
extern double pnorm(), intgrl();
extern FUNCTPTR f_alloc();
extern int free_funct();
FUNCTPTR f;
double y;

   f = f_alloc(10);
   FFPARM(f,0) = pnorm;
   y = intgrl(-5.0,x,f);
   free_funct(f);
   return(y);
}



double absc2(yjm1,rj,yj)
double yjm1,rj,yj;
{
extern FUNCTPTR f_alloc();
extern int free_funct();
extern double intgrl(),cum_norm(),absc2a();
extern double k1,k2,n_var;
FUNCTPTR f1;
double a,b,mean,std;

   f1 = f_alloc(10);
   FFPARM(f1,0) = absc2a;
   FDPARM(f1,2) = rj;
   a = intgrl(yjm1,yj,f1);
   mean = k1 * rj + k2;
   std = sqrt( k1 * n_var );
   b = (yjm1-yj)/2.
     + yj*cum_norm((yj-mean)/std) - yjm1*cum_norm((yjm1-mean)/std);
```

```
  free_funct(fl);
/*
  printf("a:%lf b:%lf yjml:%lf rj:%lf yj:%lf\n",a,b,yjml,rj,yj);
*/
  return(a - b);
}


double absc2a(f)
FUNCTPTR f;
{
extern double p_normal(),k1,k2,n_var,sqrt();
double x,y,rj,mean,std;
  x = FDPARM(f,1);
  rj = FDPARM(f,2);
  mean = k1 * rj + k2;
  std = sqrt(n_var * k1);
  y = x * p_normal(mean,std,x);
  return(y);
}


double condition(f)
FUNCTPTR f;
{
extern double fabs(),sgn();
extern double r_std,r_mean,r_inf;
double (*c)();
double x1,x2,x3,diff,last_diff,add;

  c = FFPARM(f,0);
  x1 = FDPARM(f,1);
  x2 = FDPARM(f,2);
  x3 = FDPARM(f,3);

  if( x3 == 0. ) {                  /* first time calc */
    x3 = x1 + r_inf;                      /* guess */
  }

  add = INCREMENT;
  last_diff = (*c) (x1,x2,x3);
/*
  printf("x1:%lf  x2:%lf  x3:%lf  diff:%lf\n",x1,x2,x3,last_diff);
```

```
*/
  if (fabs(last_diff) < FIDELITY) return(x3);

  x3 += add;

  diff = (*c) (x1,x2,x3);
/*
  printf("x1:%lf  x2:%lf  x3:%lf  diff:%lf\n",x1,x2,x3,diff);
*/
/*------------------------------------------------------------
  Now determine if we are going in the right direction.
----------------------------------------------------------*/

  if ( diff/last_diff > 1. ) {
    x3 -= 2. * add;
    add = -add;
    last_diff = diff;
    diff = (*c) (x1,x2,x3);
/*
  printf("x1:%lf  x2:%lf  x3:%lf  diff:%lf\n",x1,x2,x3,diff);
*/
  }

/*----------------------------------------------------------------
  Now we just keep incrementing x3 until we have it bracketed,
  determined via a sign change in diff.
-------------------------------------------------------------*/

  while ( sgn(diff) == sgn(last_diff) ) {
    if ( fabs(diff) < FIDELITY ) return(x3);
    last_diff = diff;
    x3 += add;
    diff = (*c) (x1,x2,x3);
/*
  printf("x1:%lf  x2:%lf  x3:%lf  diff:%lf\n",x1,x2,x3,diff);
*/
  }

/*------------------------------------------------------------
  We've bracketed x3 now and only monitor sign change to
  determine direction of add... It will continuously be
  divided in half since we know both bracket limits.
-------------------------------------------------------------*/
```

```
  while ( (fabs(diff) >= FIDELITY) && (fabs(add) >= FIDELITY) ) {
    add /= (sgn(diff) != sgn(last_diff)) ? -2. : 2.;
    last_diff = diff;
    x3 += add;
    diff = (*c) (x1,x2,x3);
/*
  printf("x1:%lf  x2:%lf  x3:%lf  diff:%lf\n",x1,x2,x3,diff);
*/
  }

  return(x3);
}


double find_quant(q)
QUANTPTR(q);
{
extern double abscl(), absc2(), condition(), r_mean, r_inf;
extern FUNCTPTR f_alloc();
FUNCTPTR f;
extern int free_funct();
double thr,lev;
int i,nlev;

  f = f_alloc(10);
  nlev = NLEVELS(q);

    for(i = 0; i < (nlev - 1); ++i) {
      FFPARM(f,0) = abscl;
      FDPARM(f,1) = THRESH(q,i);
      FDPARM(f,2) = LEVEL(q,i);
      FDPARM(f,3) = THRESH(q,i+1);
      THRESH(q,i+1) = condition(f);
      FFPARM(f,0) = absc2;
      FDPARM(f,1) = LEVEL(q,i);
      FDPARM(f,2) = THRESH(q,i+1);
      FDPARM(f,3) = LEVEL(q,i+1);
      LEVEL(q,i+1) = condition(f);
    }
    free_funct(f);
    thr = THRESH(q,nlev-1);
    lev = LEVEL(q,nlev-1);
    return( abscl(thr,lev,r_mean + r_inf) );
}


main()
{
extern QUANTPTR q_alloc(), q;
extern FILE *fopen();
extern int fclose(),set_initial(),print_quantizer(),zero_quant();
extern int sset_initial();
extern double sqrt(),fabs(),find_quant(),sgn();
extern double x_mean,n_mean,r_mean,x_inf,n_inf,r_inf;
extern double x_std,x_var,n_std,n_var,r_std,r_var;
extern double k1,k2;
double add,guess,diff,last_diff;
double nstd,xstd,scale;
FILE *io;
QUANTPTR qr;
char name[25];
int nlev,i,j;
  printf("Minimum Risk Quantizer (Abs - loss) Program\n");

  printf("Enter output filename:");
  gets(name);

  io = fopen(name,"w");

  printf("Enter number of quantizer levels: ");
  scanf("%d",&nlev);

  qr = q_alloc(nlev);

  nlev /= 2;

  q = q_alloc(nlev);

  printf("Enter signal mean and std: ");
  scanf("%lf %lf",&x_mean,&xstd);
  printf("Enter noise mean and std: ");
  scanf("%lf %lf",&n_mean,&nstd);

  scale = xstd;              /* Set scale to calculate on N(mean,1.) */
  x_std = 1.0;               /* then rescale {r}, {y} back out later.*/
```

```
    n_std = nstd / scale;


    x_var = x_std * x_std;
    n_var = n_std * n_std;
    r_var = x_var + n_var;
    r_std = sqrt(r_var);
    r_mean = x_mean + n_mean;

    x_inf = 5.0 * x_std;
    n_inf = 5.0 * n_std;
    r_inf = 5.0 * r_std;
/*--------------------------------------
    if(nlev < 12)
      increment = 0.1;
    else if(nlev < 45)
      increment = 0.01;
    else if(nlev < 129)
      increment = 0.001;
    else
      increment = 5.0e-8;
--------------------------------------*/
    increment = INCREMENT;
    add = increment;

    guess = x_mean + 3. * increment;

    k1 = x_var / (x_var + n_var);
    k2 = (x_mean * n_var - n_mean * x_var)
          / (x_var + n_var);

    zero_quant(q);
    sset_initial(q,guess);
    printf("Initial Quantizer:\n");
    print_quantizer(q);
    printf("\nNext:\n");
    diff = find_quant(q);
    print_quantizer(q);
    printf("\nDiff: %lf\n",diff);

    if (fabs(diff) >= FIDELITY) {
      last_diff = diff;
      guess += add;
    set_initial(q,guess);
    diff = find_quant(q);
    print_quantizer(q);
    printf("\n");

/*------------------------------------------------------------
  Now determine if we are going in the right direction.
------------------------------------------------------------*/

    if ( diff/last_diff > 1. ) {
      guess -= 2. * add;
      add = -add;
      last_diff = diff;
      set_initial(q,guess);
      diff = find_quant(q);
      print_quantizer(q);
      printf("\n");
    }

/*------------------------------------------------------------------
  Now we just keep incrementing guess until we have it bracketed,
  determined via a sign change in diff.
------------------------------------------------------------------*/

    while ( (sgn(diff) == sgn(last_diff))
             && (fabs(diff) >= FIDELITY) ) {
      last_diff = diff;
      guess += add;
      set_initial(q,guess);
      diff = find_quant(q);
      print_quantizer(q);
      printf("\n");
    }

/*---------------------------------------------------------------
  We've bracketed guess now and only monitor sign change to
  determine direction of add... It will continuously be
  divided in half since we know both bracket limits.
---------------------------------------------------------------*/

    while ( (fabs(diff) >= FIDELITY) && (fabs(add) >= FIDELITY) ) {
      add /= (sgn(diff) != sgn(last_diff)) ? -2. : 2.;
      last_diff = diff;
```

```
        guess += add;
        set_initial(q,guess);
        diff = find_quant(q);
        print_quantizer(q);
        printf("\n");
      }
    }

    for(i=0; i<nlev; ++i)
      LEVEL(q,i) = LEVEL(q,i) - x_mean;

    j = nlev;
    for(i=0; i<nlev; ++i) {
      THRESH(qr,i) = THRESH(q,0) - scale * (THRESH(q,j)-THRESH(q,0));
      LEVEL(qr,i) = x_mean - scale * LEVEL(q,j-1);
      --j;
      THRESH(qr,i+nlev) = THRESH(q,0) + scale * (THRESH(q,i)-THRESH(q,0));
      LEVEL(qr,i+nlev) = x_mean + scale * LEVEL(q,i);
    }
    THRESH(qr,0) = -10000.;
   print_quantizer(qr);
                                                /* Scale to actual stdev */
   fprint_quant(qr,io);
   fprintf(io,"%lf\n",10000.);

   free_quant(q);
   free_quant(qr);

   fclose(io);
   exit(0);
  }
```

```
/*----- Program to generate data for Risk Surfaces -----*/

#include <stdio.h>
#include <math.h>
#include <function.h>
#include <quantizer.h>

QUANTPTR q, qrisk, qunif, qmax;
double r_inf;
double x_inf;
double n_inf;
double n_mean,n_std,x_mean,x_std,r_mean,r_std;
double k1,k2,x_var,n_var,r_var;


double p_normal(mean,std,x)
double mean;
double std;
double x;
{
      extern double fabs(),exp();
      double arg;

        if (fabs(x) > (5.0 * std + mean))
          return(0.);
        else {
          arg = -(x-mean)*(x-mean)/(2.*std*std);
          return( 0.3989422804 * exp(arg) / std);
        }
}

double p_noise(n)
double n;
{
      extern double n_mean, n_std, p_normal();
          return(p_normal(n_mean,n_std,n));
}
```

```
double p_x(x)
double x;
{
      extern double x_mean, x_std, p_normal();
        return(p_normal(x_mean,x_std,x));
}


/*-----------------------------------------------------------------
  Observation probability density function.                          :
   Defined as the convolution of p(x) and p(n) at                    :
   the observation point r.                                         */


double p_r(f)
FUNCTPTR f;
{
      extern double p_normal(), r_mean, r_std;
      double r;

        r = FDPARM(f,1);
        return(p_normal(r_mean,r_std,r));
}



double dist(f)
FUNCTPTR f;
{
double r,y;
extern double p_r();
  r = FDPARM(f,1);
  y = FDPARM(f,2);
  r = (r - y) * (r - y) * p_r(f);
  return(r);
}
```

```
double inner(f)
FUNCTPTR f;
{
extern double p_noise(),p_x();
double x,y,r;

  x = FDPARM(f,1);
  y = FDPARM(f,2);
  r = FDPARM(f,3);
  return( (x - y) * (x - y) * p_noise(r - x) * p_x(x) );
}


double innerisk(f)
FUNCTPTR f;
{
extern double p_r();
extern double x_var,n_var,r_var,k1,k2;
double value,r,y,c;
  r = FDPARM(f,1);
  y = FDPARM(f,2);
  c = (x_var * n_var) / r_var;
  value = (k1 * r + k2 - y) * (k1 * r + k2 - y);
/*  printf("c:%lf k1:%lf k2:%lf squared term:%lf\n",c,k1,k2,value);
*/
  value += c;

  return(value * p_r(f));
}


double calc_risk(q)
QUANTPTR q;
{
extern double innerisk(),integral(),r_inf,r_mean;
double rl,ru,y,risk,pos,neg,two_mean;
int i,num_levels;
extern FUNCTPTR f_alloc();
FUNCTPTR f;

  f = f_alloc(4);
  FFPARM(f,0) = innerisk;
  risk = 0.;
  num_levels = NLEVELS(q);

  if((NLEVELS(q) % 2) == 0) {           /* even */
    for(i = 0; i < num_levels - 1; ++i) {
      y = LEVEL(q,i);
      rl = THRESH(q,i);
      FDPARM(f,2) = y;
      ru = THRESH(q,i+1);
      pos = integral(rl,ru,f);
      risk += pos;
    }
    y = LEVEL(q,num_levels-1);
    FDPARM(f,2) = y;
    rl = THRESH(q,num_levels-1);
    ru = r_mean + r_inf;
    pos = integral(rl,ru,f);
    risk += pos;
  }
  else
    printf("Section not implemented for odd number of levels\n");

  free_funct(f);
  return(risk);
}



double calc_mrisk(q)
QUANTPTR q;
{
extern double dist(),p_r(),integral(),r_inf,r_mean;
extern double x_var,n_var,r_var,k1,k2;
double rl,ru,y,c,d,p,risk,pos,neg,two_mean;
int i,num_levels;
extern FUNCTPTR f_alloc();
FUNCTPTR f1, f2;

  f1 = f_alloc(4);
  f2 = f_alloc(4);
  FFPARM(f1,0) = p_r;
  FFPARM(f2,0) = dist;
  c = (x_var * n_var) / r_var;
  risk = 0.;
  num_levels = NLEVELS(q);

  if((NLEVELS(q) % 2) == 0) {           /* even */
    for(i = 0; i < num_levels - 1; ++i) {
      y = LEVEL(q,i);
      rl = THRESH(q,i);
      FDPARM(f2,2) = (y - k2) / kl;           /* convert to max level
*/
      ru = THRESH(q,i+1);
      p = integral(rl,ru,fl);
      d = integral(rl,ru,f2);
      pos = c * p + kl * kl * d;
/* printf("c:%lf rl:%lf ru:%lf p:%lf kl:%lf d:%lf\n",c,rl,ru,p,kl,d);
*/
      risk += pos;
    }
    y = LEVEL(q,num_levels-1);
    FDPARM(f2,2) = (y - k2) / kl;           /* convert to max level */
    rl = THRESH(q,num_levels-1);
    ru = r_mean + r_inf;
    p = integral(rl,ru,fl);
    d = integral(rl,ru,f2);
    pos = c * p + kl * kl * d;
/* printf("c:%lf rl:%lf ru:%lf p:%lf kl:%lf d:%lf\n",c,rl,ru,p,kl,d);
*/
    risk += pos;
  }
  else
    printf("Section not implemented for odd number of levels\n");

  free_funct(fl);
  free_funct(f2);
  return(risk);
}



double db(x)
double x;
{
      extern double log10();
        return(10. * log10(x));
}
```

```
double dbinv(x)
double x;
{
      extern double pow();
        return( pow(10., 0.1 * x) );
}


main()
{
      extern QUANTPTR q_alloc(), q, qrisk, qunif, qmax;
      extern double p_r();
      extern double r_var, kl, k2, x_inf, n_inf, r_inf;
      extern double fabs(), log10(), pow(), sqrt();
      extern double calc_risk(),calc_mrisk(),db(),dbinv();
      extern int free_quant();
      extern double x_mean,x_std,n_mean,n_std,r_mean,r_std;
      double thresh[128],level[128],delta;
      extern double x_var,n_var;
      FILE *io, *maxfile;
      double area, last_area, (*p) ();
      double sn_ratio,risk;
      double rlower,rupper,incr,nstd;
      double xmlower,xmupper,xmincr,nmlower,nmupper,nmincr;
      int i,j,num,num_levels;

      printf("Risk Curve Generation Program\n");

      io = fopen("RISKPLOT OUT B","w");

      maxfile = fopen("MAX DATA B","r");

      p = p_r;

      printf("Enter range and increment of S/N ratio.\n");
      printf("(-10dB to 15dB by 5dB --> -10. 15. 5.)\n");
      scanf("%lf %lf %lf",&rlower,&rupper,&incr);

      printf("Enter range and increment of signal mean: ");
      scanf("%lf %lf %lf",&xmlower,&xmupper,&xmincr);
```

```
        if(xmincr <= 0.) xmincr = 1.;

        printf("Enter range and increment of noise mean: ");
        scanf("%lf %lf %lf",&nmlower,&nmupper,&nmincr);

        if(nmincr <= 0.) nmincr = 1.;

        printf("Enter number of levels: ");
        scanf("%d",&num);

        for(i=2; i<=num; ++i)
          for(j=0; j<i; ++j)
            fscanf(maxfile,"%lf %lf",&thresh[j],&level[j]);

        fclose(maxfile);

        for(j=num-1; j>0; --j)
          thresh[j] = thresh[j-1];

        thresh[0] = -INFINITY;
        thresh[num] = INFINITY;
        level[num] = 0.;

        x_std = 1.0;

        qmax = q_alloc(num);          /*  Max-Lloyd Std. Normal  */
        for(i = 0; i < num; i++) {    /*  risk surf only makes sense for
*/
          THRESH(qmax,i) = thresh[i]; /* x_mean = 0 */
          LEVEL(qmax,i) = level[i];
        }


        n_mean = nmlower;
        while(n_mean <= nmupper) {
          x_mean = xmlower;
          while(x_mean <= xmupper) {
          nstd = rlower;
            while(nstd <= rupper) {
            n_std = sqrt(1. / dbinv(nstd));
              r_mean = x_mean + n_mean;
              r_var = x_std * x_std + n_std * n_std;
              r_std = sqrt(r_var);
```

```
    r_inf = 5.0 * r_std;
    x_inf = 5.0 * x_std;
    n_inf = 5.0 * n_std;

    q = q_alloc(num);
    qrisk = q_alloc(num);
    qunif = q_alloc(num);

    x_var = x_std * x_std;
    n_var = n_std * n_std;

    k1 = x_var / (x_var + n_var);
    k2 = (x_mean * n_var - n_mean * x_var) / (x_var + n_var);

    delta = 2. * r_inf / num;

    THRESH(qunif,0) = 2. * r_mean - r_inf;
    LEVEL(qunif,0) = THRESH(qunif,0) + delta / 2.;

    for(i = 1; i < num; ++i) {
      THRESH(qunif,i) = THRESH(qunif,i-1) + delta;
      LEVEL(qunif,i) = LEVEL(qunif,i-1) + delta;
    }

    for(i = 0; i < num; ++i) {
      THRESH(q,i) = r_std * thresh[i] + r_mean;
      LEVEL(q,i) = r_std * level[i] + r_mean;
      THRESH(qrisk,i) = THRESH(q,i);
      LEVEL(qrisk,i) = k1 * LEVEL(q,i) + k2;
    }

    THRESH(qmax,0) = r_mean - r_inf;
    THRESH(q,0) = r_mean - r_inf;
    THRESH(qrisk,0) = THRESH(q,0);

    sn_ratio = db(x_var/n_var);
    fprintf(io,"%lf %lf %lf %lf",x_mean,x_std,n_mean,n_std);
    printf("%lf %lf %lf %lf",x_mean,x_std,n_mean,n_std);

printf("Max-Lloyd Std. Normal:\n\n");
    print_quantizer(qmax);
printf("\n");
```

```
          risk = calc_risk(qmax);
              fprintf(io," %lf",risk);
              printf(" %lf",risk);

          printf("Max-Lloyd Quantizer:\n\n");
              print_quantizer(q);
          printf("\n");
          risk = calc_risk(q);
              fprintf(io," %lf",risk);
              printf(" %lf",risk);
/*        risk = calc_mrisk(q);
              fprintf(io," %lf",risk);
              printf(" %lf",risk);
*/

          printf("\n\nMinimum-Risk Quantizer:\n\n");
              print_quantizer(qrisk);
          printf("\n");
          risk = calc_risk(qrisk);
          printf("\n");
              fprintf(io," %lf",risk);
              printf(" %lf\n",risk);

/*        risk = calc_mrisk(qrisk);
              fprintf(io," %lf",risk);
              printf("mrisk: %lf\n",risk);
*/
/*------ calc_risk and calc_mrisk produced exactly same values ---*/
          printf("\n\nUniform Quantizer:\n\n");
             print_quantizer(qunif);
         printf("\n");
         risk = calc_risk(qunif);
         printf("\n");
             fprintf(io," %lf\n",risk);
             printf(" %lf\n",risk);
              free_quant(q);
              free_quant(qrisk);
              free_quant(qunif);
              nstd += incr;
            }
            x_mean += xmincr;
          }
          n_mean += nmincr;
```

```
		}
		fclose(io);
		exit(0);
	}
```